Diffusion model construction and parameter adjustment Playbook (strictly based on 6 papers)

Engineering structured summary strictly based on 6 papers (publication layout edition)

Document Type: Structured Technology Review and Practice Manual (Journal Available Version)

Editor: Beibei Jing

Version: 2026-02-14

Scope: Only summarize and connect the design choices and experiences appearing in the following 6 papers; no additional parameter adjustment conclusions outside the papers will be introduced.

Writing Constraints: Strictly 4 paragraphs per entry: `How / Why / Common Issues / Fix`.

Contents

summary

Within the strictly limited scope of six documents, this article systematically sorts out the key design variables of diffusion model training and sampling, and provides reproducible implementation and parameter adjustment paths in an itemized structure. The content covers data processing, model structure, loss and optimization, sampling and evaluation, and summarizes comparative suggestions on few-step inference and covariance learning. The manuscript emphasizes reproducibility and reviewability, and all entries are marked with source page numbers for review.

keywords

Diffusion model; sampler; preconditioning; variance/covariance learning; evaluation protocol; reproducibility

Contribution and scope statement

- Under unified symbols and reference standards, the key design choices and experiences that can be reproduced in P1-P6 are summarized to form a reviewable implementation list.

- A closed-loop parameter adjustment path of "phenomenon-positioning-modification-review" is provided, as well as a minimal experimental protocol coupled with the evaluation link.

- Make it clear that the conclusion boundaries of the six papers will not be exceeded, and provide reviewer-verifiable record requirements for data, evaluation, and sampler calibers.

Reading Guide (Readers, Usage, Quotations and Notations)

- Readers: Engineering and scientific research readers who do diffusion model training/sampling implementation and system ablation (familiar with basic probability and deep learning training by default).

- Usage: Read in a closed loop of "Phenomenon-Location-Change-Review"; Chapter 5 (Visualization and Fine-tuning) is used to include the sampling link in the parameter adjustment panel.

- Citation: `Ref: [P?, p.?]` at the end of each entry title indicates the source; `p.?` corresponds to the page number of the paper PDF (aligned with the `=== PAGE ? ===` tags of `Books/text/*.txt`).

- Notation: `T` total diffusion steps, `t` time steps, `alpha_bar_t` cumulative noise coefficient, `sigma` continuous noise scale (EDM), `NFE` number of network evaluations during sampling, `FID/NLL` common evaluation indicators.

Six papers (the only basis for this manual)

- [P1] Alex Nichol, Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672, 2021.

- [P2] Jiaming Song, Chenlin Meng, Stefano Ermon. Denoising Diffusion Implicit Models. arXiv:2010.02502, 2021.

- [P3] Tero Karras, Miika Aittala, Timo Aila, Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. arXiv:2206.00364, NeurIPS 2022.

- [P4] Hang Chen, Qian Xiang, Jiaxin Hu, Meilin Ye, Chao Yu, Hao Cheng, Lei Zhang. Comprehensive exploration of diffusion models in image generation: a survey. Artificial Intelligence Review, 58:99, 2025.

- [P5] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, Yang You. Dynamic Diffusion Transformer. ICLR 2025.

- [P6] Zijing Ou, Mingtian Zhang, Andi Zhang, Tim Z. Xiao, Yingzhen Li, David Barber. Improving Probabilistic Diffusion Models with Optimal Diagonal Covariance Matching. arXiv:2406.10808, ICLR 2025.

Citation format: Write `Ref: [P? p?]` in the title of each entry.

0) First principles (only as long as it is enough)

0.0 Getting started example in this chapter: Use `L_simple` to run through the minimum closed loop of " one-step training + one-step visualization" Citation: P1 p2

How	This is the smallest closed loop you can write a training script for, and every symbol is defined in this section. 1) Take a batch of real images x_0 and normalize to [-1,1] (e.g. $x = (x - 0.5) / 0.5$). 2) Sample time step t~Uniform({0⋯T − 1}) and noise $\varepsilon \sim N(0, I)$. 3) Compose x_t: $x_t = \sqrt{(\bar{\alpha}_t)} * x_0 + \sqrt{(1 - \bar{\alpha}_t)} * \varepsilon$ with margins. 4) Network forward: $\varepsilon_\theta = \text{model}(x_t, t)$. 5) Training target: $L_{simple} = E \| \varepsilon - \varepsilon_\theta \|^2$, back-transmit and update parameters once. 6) Do a visualization every fixed number of steps: fix the same set of x_T and use a deterministic sampler to generate a grid plot (see 5.0).
Why	P1 points out that this goal is equivalent to the reweighted form of VLB and is more stable in practice; as long as this closed loop is first run through, all subsequent "sampler changes/variance learning/network changes" will have a comparable baseline.
Common Issues	The training loss decreases but the sampling looks like noise. The common reason is that the $\bar{\alpha}_t$/broadcast dimension/sampling mean variance term is written incorrectly, or the normalization caliber is inconsistent during visualization.
Fix	1) Precompute $\bar{\alpha}_t$ and write unit tests: it decreases monotonically with t, and is close to 1 in $t = 0$, and close to 0 in $t = T - 1$. 2) Fixed x_T for consistency diagnosis: only change the number of sampling steps/trajectories, and the high-level semantics should be basically consistent (see 5.2).

0.1 What exactly is the diffusion model learning (parameterization of the inverse distribution) Citation: P1 p2, P6 p1-p2

How	1) Write the forward chain clearly: Define stepwise Gaussian noise $q(x_t \mid x_{t-1})$ so that x_T approximates $N(0, I)$. 2) Precomputed and cached: β_t, $\alpha_t = 1-\beta_t$, $\bar{\alpha}_t = \prod_{i=0..t} \alpha_i$ (used for both training and sampling). 3) Select the caliber of reverse parameterization: use a network to represent $p_\theta(x_{t-1} \mid x_t) = N(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, and decide whether you "learn only the mean/noise" or "learn both the mean + variance/covariance" (see 2.7/2.8/3.2/3.6/3.7). 4) When sampling, press $t = T \cdots 1$ to gradually denoise from x_T to x_0; if you want to sample in fewer steps, treat the "sampling link" as an independent module and include it in the parameter adjustment closed loop (see Chapter 5).
Why	The true posterior $q(x_{t-1} \mid x_t)$ is not available relying on the full data distribution; it can only be approximated (mean/covariance) with a learnable Gaussian to gradually restore from noise to data.
Common Issues	When only learning the mean and covariance with a fixed heuristic, the quality/likelihood will drop significantly when the number of sampling steps is very small (small T or skip steps); the training loss decreases but the sampling link may still "look like noise".
Fix	1) Few-step first only changes the sampling but not the training: Use DDIM subsequence/deterministic sampling to draw the "number of steps-quality curve" (see 5.1). 2) Then change the numerical integration and time points: use EDM/Heun, etc. to suppress the sampling error (see 5.3). 3) If the goal is to maintain 5/10/20-step quality and likelihood: Use Σ_θ as an explicit design variable and introduce learning variance (P1) or OCM (P6) (see 3.6-3.7, 5.7-5.8).

0.2 Why MSE with "prediction noise" is commonly used in training Citation: P1 p2

How	1) Each batch randomly samples time step $t \sim \text{Uniform}(\{0 \cdots T-1\})$, and samples $\varepsilon \sim N(0, I)$. 2) Use marginal one-time synthesis of x_t: $x_t = \sqrt{(\bar{\alpha}_t)} * x_0 + \sqrt{(1 - \bar{\alpha}_t)} * \varepsilon$ (training samples do not need to be progressively noisy). 3) Feed x_t and t to the network and predict the noisy $\varepsilon_\theta(x_t, t)$. 4) Train with $L_{simple} = E \| \varepsilon - \varepsilon_\theta \|^2$.
Why	The paper points out that this objective can be regarded as a reweighted form of VLB (and more like score matching), and in practice the sample quality is better than directly optimizing VLB.
Common Issues	L_{simple} has no learning signal for $\Sigma_\theta(x_t, t)$; if you also want to learn variance/covariance, this goal alone is not enough.
Fix	1) To learn variance: enable $L_{hybrid} = L_{simple} + \lambda * L_{vlb}$ with variance parameterization (see 2.8/3.2/3.3). 2) If you want less-step quality: Change the sampling link (DDIM/EDM/Heun) first, and then consider covariance learning (OCM) (see 5.1-5.4, 3.6-3.7).

0.3 “ Training is constant, sampling is variable ” : DDIM generalizes the reverse process to non-Markovian Citation: P2 p6-p7

How

1) The training phase remains unchanged: T = 1000 and the original training target are still used to train the model.
2) Subsequence selection τ Select one of the two clear rules in P2 (write down the formula and do not leave it to the reader to guess):
- Linear: τ_i = floor(c * i), choose c to make tau_{N − 1} close to T.
- Quadratic: τ_i = floor(c * i^2), choose c so that tau_{N − 1} is close to T.
- Empirical selection of P2: use quadratic for CIFAR-10 and linear for the rest of the data sets (FID is slightly better).
3) Set randomness η (definition of P2): η = 0 is deterministic DDIM; η = 1 corresponds to the original DDPM generation process.
4) The sampling update formula is implemented according to the closed formula of P2 (each step can be reviewed):
- x_{tau_{i − 1}}(η) = √(alpha_{tau_{i − 1}}) * ((x_{τ_i} − √(1 − alpha_{τ_i}) * ε_θ(x_{τ_i}, τ_i)) / √(alpha_{τ_i})) + √(1 − alpha_{tau_{i − 1}} − sigma_{τ_i}(η)^2) * ε_θ(x_{τ_i}, τ_i) + sigma_{τ_i}(η) * ε
- sigma_{τ_i}(η) = η * √((1 − alpha_{tau_{i − 1}}) / (1 − alpha_{τ_i})) * √(1 − alpha_{τ_i} / alpha_{tau_{i − 1}})
5) The fixed evaluation protocol compares 10/20/50/100 step output and forms a "less-step acceleration curve" (see 5.1/5.5/5.6).

Why

P2 explains that DDIM and DDPM share the same training marginal distribution, but the reverse generation process can choose a family of non-Markov processes; therefore "train once, inference can switch at different speed-quality points".

Common Issues

Mistaking "slow generation" for insufficient training; or mistakenly thinking that you need to retrain a model with a smaller T to speed up.

Fix

1) First make sure that only τ / η is changed under "the same model + the same initial x_T set", otherwise the curves will not be comparable (see 5.2).
2) If the few steps are still like noise: first check whether $\bar{\alpha}_t$ is consistent with the update implementation; then consider changing to EDM/Heun or introducing variance/covariance learning (see 5.3-5.4, 3.2/3.7).

0.4 “ Sampling is numerical integration ” : EDM breaks down the design space into operable knobs
Citation: P3 p1, p45

How	1) Think of generative explicitation as "denoising/fractional learning on a continuous noise scale". 2) Split the implementation into 5 knobs that can be turned on and off independently: preconditioning, training noise distribution, loss weighting, sampler (deterministic/random), discrete time point and solver (Heun/Euler). 3) First fix "deterministic sampler + fixed NFE" as the alignment baseline (see 5.6), and then ablate each knob one by one.
Why	
Common Issues	When only changing one factor (such as changing only the solver or only changing the weight), the returns are unstable or even worse, because the other knobs still do not match it.
Fix	1) Only move one "knob set" per change and write it to the configuration record (data/network/loss/sampler/NFE/time point). 2) If the income is unstable: Return to the deterministic sampler and fixed evaluation sample size, first eliminate evaluation fluctuations and then make a judgment (see 5.5-5.6).

1) Data processing

1.0 Getting started with this chapter: CIFAR-10 32x32 "Reproducible Data Protocol" Citation: P1 p11; P4 p37

How	
Why	
Common Issues	Inconsistent normalization calibers (for example, [0,1] for training, [-1,1] for sampling/visualization) will lead to sample color cast or abnormal contrast; mixed training set/test set will distort the evaluation.
Fix	1) Write "normalized to [-1,1]" as the only caliber for the entire link (training, sampling, and visualization are consistent). 2) After each change of data filtering/enhancement strategy, first sample and save a grid chart as the " acceptance sample" of the data protocol (see 5.0).

1.1 Select dataset and resolution (and cropping/downsampling) Citation: P1 p3,p11; P2 p6; P3 p45

How	1) Clarify the task and target resolution: which output is to be generated, 32x32/64x64/256x256, etc. 2) Choose a "paper-comparable" benchmark data set as a starting point: CIFAR-10 (32x32), ImageNet-64, CelebA (64x64), LSUN (256x256). 3) Write the preprocessing as a deterministic protocol and fix it: center crop to square + area downsample to target resolution + normalized range (and record random seed and library version). 4) If the final task is more difficult (higher resolution/more complex distribution): First run through the training-sampling closed loop on the low-resolution benchmark, and then migrate and upgrade.
Why	Resolution and diversity determine the difficulty of training and the risk of overfitting; P1 directly uses ImageNet-64 as a compromise between "diversity and trainability" for system ablation.
Common Issues	It is more difficult to optimize when the resolution/diversity is higher, requiring longer training and stronger computing power; low resolution is easily misjudged as "the model works" but the migration fails.
Fix	1) After fixing the sampler and evaluation protocol (number of steps/NFE/sample size/whether to use EMA), conduct a control experiment (see Chapter 5). 2) Before migrating to higher resolutions, review whether the preprocessing protocol is consistent (differences in cropping and downsampling will directly change the training distribution).

1.2 Data set limitations and data scarcity (research planning level) Citation: P4 p37

How

You can judge "scale/quality/bias" without reading the paper, and it is completely reproducible.
1) First, classify the "scale" (use the recurring magnitudes in the 6 papers as anchor points to facilitate unambiguous writing of papers/reports):
- Level 5e4: CIFAR-10 level (training set 50K).
- Level 1e6: ImageNet level (common practice is million-level images; P1/P3 ImageNet-64 are all typical difficulties of this level).
- Level 1e9: text-to-image common “ billions of (text, image) pairs ” (P4 uses this to explain the realistic premise of T2I mass jump).
2) Then implement the "high-quality" of P4 into a "Data Quality Inspection Statistics Table" that must be delivered (only for statistics, no additional "threshold conclusions" will be introduced here):
- The sampling caliber is hard-coded: K = 50, 000 samples are uniformly sampled for quality inspection (50K is selected to be of the same magnitude as P3's FID evaluation sample size, so as to facilitate the alignment of statistical caliber); if the total amount is less than 50K, the full amount is inspected and K = | D | is written.
- Availability (bad samples): decode samples one by one, statistics $\text{corrupt}_{\text{count}}$ / K; also record the distribution of failure reasons (file damage/decoder not supported/size is 0, etc.).
- Resolution and compression quality: Statistics of width and height distribution, short side distribution, (if any) JPEG quality/file size distribution; quantile table (Q1/median/Q3/p95) is given.
- Duplication/near duplication: Statistics on the proportion and caliber of "complete duplication" (byte-level hash) and "near duplication" (perceived hash or embedding similarity); you must clearly write down the algorithm and threshold you use (otherwise it cannot be reproduced).
- Task Alignment Noise: Only fill in when you are doing conditional generation/labeling tasks:
- text-image: Randomly check/manually review the proportion of mismatched samples, and record the mismatch type (overly broad description/wrong reference/irrelevance, etc.).
- class label: Randomly check the proportion of wrong labels and the pairs of common wrong labels (such as cat↔dog).

3) Finally, do "bias risk disclosure" according to the dimensions reminded by P4 (not to let you correct the deviation immediately, but to let readers know where your data coverage gaps are):
- Do at least one stratified statistics or random inspection according to the three axes of language / ethnicity / gender (depending on whether your task contains people-related content); the output conclusion must include: coverage gap + possible impact generation bias type + whether you did stratified evaluation during the evaluation.

Why

P4 clearly pointed out that the rapid development of diffusion models is strongly related to "large-scale high-quality data"; however, many sub-tasks will encounter data scarcity and data bias, which will lead to quality and fairness issues.

Common Issues

Small data leads to overfitting/poor generalization; data bias leads to skewed results and triggers downstream risks.

Fix

1) First, use observable signals to confirm "whether scarcity leads to overfitting": fix the sampler and NFE, save the snapshot with training and draw the FID/NLL curve (see 5.6).
2) If scarcity is confirmed: Prioritize regularization such as non-leaky augmentation (see 1.3) and use a deterministic sampler to lock the evaluation link.
3) If it is more like a "quality/bias problem" than a "scale": Go back to the 1.5 publication-level data card and complete the random inspection statistics, and then talk about model parameter adjustment.

1.3 Non-leaky augmentation as regularizer Citation: P3 p45

How	1) Select a group of enhancements (geometry/color, etc.) and record the random parameters of the enhancement as a. 2) During training: Apply enhancement to the input image, and input a as a condition to the network (let the model "know what you enhanced"). 3) It is agreed that a = 0 means "no enhancement"; a = 0 is fixed during sampling/evaluation to ensure that the generated distribution does not have traces of enhancement. 4) According to P3 ’ s implementation strategy: disable dataset x-flips, obtain enhancement benefits only through non-leaky augmentation, and avoid generating mirrored text/Logo. 5) Write the "enhancement enablement probability" as an explicit hyperparameter and record it: P3's training table gives an example of augment probability as CIFAR-10: 12%, FFHQ/AFHQ: 15% (for its improved configuration comparison).
Why	P3 explained that the idea is to treat enhancement as an auxiliary task/regular, but avoid enhancement "leakage to the generated image" through conditional input; you can even use 100% x-flip to obtain enhancement benefits without causing mirror text/logo to appear in the generation.
Common Issues	
Fix	1) If you are currently using "direct enhancement without feeding parameters": Change to non-leaky caliber and do a comparative evaluation again. 2) If it is found that the enhancement makes the FID jitter: reduce the intensity of the enhancement or keep only a subset that is not harmful to the task, and keep the sampling convention of a = 0 unchanged.

1.4 When there is no need to enhance regularity: based on "whether you are worried about overfitting"
Citation: P3 p45-p46

How	1) First fix the evaluation link: use deterministic sampler + fixed NFE + fixed sample size (see 5.6). 2) Use the training process signal to judge overfitting: If the training loss decreases but the FID continues to get worse in the later stage under a fixed sampler, it is regarded as an overfitting/generalization degradation signal (P3). 3) Use the generated diversity signal to review: If the sample is more like the nearest neighbor of the training set/the structure is repeated, it can be combined with the coverage perspective of precision/recall to make a judgment (P1). 4) If there is no obvious overfitting signal: It is also true without turning on the enhanced regularization (P3 explicitly adopts this strategy in the ImageNet-64 configuration).
Why	
Common Issues	Blindly adding enhancement to big data that is not easy to overfit may only increase the burden of parameter adjustment; or cause side effects that are inconsistent with the target distribution.
Fix	1) Enable enhancement only after overfitting is confirmed; and non-leaky augmentation is preferred to avoid distribution leakage (see 1.3). 2) After enhancement is turned on, continue to use the same evaluation link to track whether FID/NLL has been steadily improved. Don't just look at the training loss.

1.5 Minimum “ publication-grade ” checklist of data quality (for reproduction and review) Citation: P1 p11; P3 p45; P4 p37

How	Publisher/reviewer can review the caliber. 1) State the source and permissions: data source, license/scope of use, whether there are copyright/privacy risks (P4). 2) Specify size and distribution: number of samples, resolution distribution, (if applicable) category/text length distribution, apparent long tails and gaps. 3) State the quality inspection and filtering: sampling grid visualization; damaged/unreadable ratio; duplicate/near-duplicate caliber and ratio; watermark/low-definition/mismatch processing strategy. 4) State the preprocessing protocol: parameters of center crop + area downsample; normalization caliber; random seed; code and library version. 5) Specify the training-sampling protocol: noise schedule/noise distribution (such as P3's $p_{train}(\sigma)$), sampler/solver and fixed NFE (see 5.6). 6) State the biased random inspection: at least provide a hierarchical statistics or random inspection conclusion and risk description for dimensions such as language/ethnicity/gender (P4).
Why	
Common Issues	Only writing "we did cleaning/filtering" but not giving statistics, resulting in non-reproducibility; lack of pre-processing details (inconsistent cropping/downsampling/normalization caliber) leading to deviations in reproducible experiments; undisclosed data bias leading to systematic skewing of generated results.
Fix	1) Use data cards and statistical tables as a pre-training threshold: before any major training, run a quality inspection/statistics and archive the results. 2) If it is found that "parameter adjustment seems to be invalid": first check whether the preprocessing caliber is inconsistent or the distribution drift is caused by data quality/bias, and then continue to modify the model.

1.6 Dataset portrait: Provide “ reviewable input ” for trunk selection Citation: P4 p37; P1 p11; P3 p45-p46

How

Before selecting the backbone, compress the data set into a reviewable "portrait table" (written into the experiment log/appendix), so that subsequent selection and control experiments have common input.
1) Basic information: $|D|$ (number of samples), modality (unconditional/category condition/text-image), permission and sensitive risk points (P4).
2) Resolution and preprocessing protocol: original resolution distribution, target resolution $H \times W$, cropping/downsampling methods and parameters (see 1.1/1.5).
3) Quality and Noise: Damaged/Unreadable Ratio, Compression Artifacts/Watermark Ratio, Duplicate/Near-Duplicate Ratio and Aperture (see 1.2/1.5; P4).
4) Scale binning (only for description and comparison, no threshold conclusion is introduced): Refer to the discussion in P4 to mark the scale as 5e4 / 1e6 / 1e9 and indicate the sampling diameter (see 1.2).
5) Overfitting risk signal (used to determine regularization and enhancement, not directly determine the backbone): First run the baseline with a fixed sampler and a fixed NFE, and observe whether the FID continues to deteriorate (P3) and the coverage degrades (P1) in the later stages of training (see 1.4/5.6).
6) Goal Reasoning Constraints: The number of steps/NFE panels to plan to report on (e.g. 5/10/20/50/100 steps or fixed NFE), and whether fewer-step reasoning is necessary (see Chapter 5, P2/P3/P6).

Why

P4 emphasizes that data scale/quality/bias will significantly affect the generation quality and risk; at the same time, the large number of ablation of P1/P3 shows that "optimal structure and regular configuration" are strongly coupled with resolution and overfitting degree. Without a portrait table, trunk selection and subsequent changes are difficult to attribute and reproduce.

Common Issues

Start testing directly from the network structure, but the data preprocessing/quality/scale and inference constraints are not locked, causing the difference in "changing the backbone/changing the sampler" to be masked by the change in data caliber.

Fix

1) First freeze the 1.5 data protocol and 5.6 evaluation point selection protocol, and reuse the same portrait table and evaluation panel for any backbone comparison.
2) If the data portrait itself is unstable (the filtering/clipping strategy changes frequently), first converge the data protocol to a stable version and then perform backbone comparison.

1.7 From portrait to trunk candidate set: Prioritize matching of "paper anchor task" Citation: P1 p11; P3 p45-p46; P5 p17; P4 p37

How

Under the constraint of "only relying on P1-P6" in this manual, the backbone selection does not pursue generalization rules. Instead, it uses "the anchor task closest to the paper experiment" to give a reproducible starting baseline, and then uses discrete gears to make the minimum sweep point.
1) First map the data set to the nearest anchor point by resolution/task:
- 32×32 Unconditional (or weakly conditional) → Using CIFAR-10 as the anchor point, the backbone preferentially uses the U-Net gear and regular grid of P1 (see 2.1/4.5) or uses the improved training/network configuration of P3 as the anchor point starting point (see the 4.5 Appendix Table).
- 64×64 unconditional/category conditions → Using ImageNet-64 as the anchor point, the backbone preferentially uses P1's ImageNet-64 U-Net configuration table as the "reproducible default" (see 2.0); if implemented according to the P3 system, start with its ImageNet related network/training table as the anchor point (see the appendix table).
- 256×256/512×512 and using the DiT system → Only when the setting is similar to your task (such as ImageNet 256/512), use the DiT-S/B/XL discrete gear of P5 as the backbone starting point (see 2.6*/Appendix A.1).
2) If the target data set does not fall at the above anchor resolution: first make an "anchor resolution pilot" (uniformly cropped/downsampled to one of 32/64/256 according to 1.1), use the same evaluation protocol to compare the backbone trend; then migrate the most stable backbone to the target resolution for review (no optimality conclusions outside the paper will be introduced here).
3) Conditional information determines the "conditional injection form" rather than directly determining the backbone paradigm: scale-and-shift conditional injection is written as a fixed implementation according to P1 (see 2.3) to ensure that the conditional paths are consistent when comparing different backbones.

Why

P1/P3/P5 respectively provide reproducible backbone scale and configuration tables on their anchor tasks; in the absence of unified selection conclusions across data sets, "match anchor points first and reproduce the baseline" can maximize comparability and reviewability, and is also in line with P4's requirements for reproducible data protocols and risk disclosure.

Common Issues

"Self-made structure/self-made scale" on new data sets, and changing preprocessing and samplers at the same time, making it impossible to determine whether the backbone, training, sampling, or data caliber cause the effect change.

Fix

1) In the backbone comparison stage, only "the backbone and its scale level" are allowed to be changed, and the rest (data protocol, training duration, sampler/NFE, and evaluation sample size) are all frozen (see 5.5-5.6).
2) The comparison is only performed within discrete gears: U-Net presses the C = 64 / 96 / 128 / 192 gear of P1, DiT presses the S/B/XL gear of P5; avoid introducing non-comparable intermediate scales.

1.8 Minimum sweep point after trunk selection: “ Discrete gear + linkage hyperparameter ” Citation: P1 p8, p11; P3 p45; P5 p17

How

After selecting a backbone family, use the discrete scale gear given in the paper to make the minimum sweep point, and link the key hyperparameters according to the caliber of the paper to obtain a reviewable "scale-quality" curve.
1) If you use P1 style U-Net: only scan within the fourth level of C = 64 / 96 / 128 / 192, and link lr according to the learning rate scaling rule of P1 (see Appendix A.6); the remaining structural fields (downsampling times, resblocks, attention resolution) are first fixed to the anchor configuration (see 2.0-2.2).
2) If using P5 style DiT: Only scan within the three levels of DiT-S/B/XL, and use the key fields of its fine-tuning/training protocol (warmup, batch, λ grid, etc.) as a reproducible starting point (see Appendix A.1-A.2).
3) If you use the improved configuration of P3: the trunk/training/sampling knobs are strongly coupled, first reproduce its configuration group as an anchor point, and then perform the ablation of "only changing one knob group at a time" (see 0.4/4.5/5.3-5.4).
4) Point selection and reporting: According to 5.6, use the fixed deterministic sampler and fixed NFE to select the optimal snapshot, and simultaneously report the FID/NLL/coverage indicator panel under the backbone gear (see 5.5-5.6).

Why

P1/P5 clearly demonstrates the reproducible comparison method of "scale gear + linked hyperparameters"; P3 emphasizes configuration coupling, and arbitrary dispersion will destroy the stability of reproduction and conclusions. Using discrete gears to scan points can avoid introducing unnecessary degrees of freedom into experiments that cannot be reviewed by reviewers.

Common Issues

When scanning the scale, only the backbone is changed without changing the learning rate/training protocol, causing changes in convergence properties to be mistaken as "the backbone is better/worse"; or the configuration group of P3 is broken up and a single hyperparameter is changed, leading to unstable conclusions.

Fix

1) Write the "scale gear → linkage super parameters" rule into the configuration system and force log printing to avoid manual overwriting during the experiment.
2) If resources are limited, first conduct the minimum gear comparison (for example, U-Net: C=64 vs 128; DiT: S vs B), confirm the trend under the fixed evaluation protocol, and then expand the scanning points.

2) Model building

2.0 Getting started example in this chapter: Directly reproduce P1's ImageNet-64 U-Net architecture configuration table Citation: P1 p11

How	Treat this paragraph as a "configuration file", and readers can directly reproduce it to achieve the same backbone. 1) Input and output resolution: 64x64. 2) Downsampling chain: 4 times downsampling, the resolution sequence is 64 -> 32 -> 16 -> 8 -> 4. 3) Number of residual blocks per downsampling stage: 3 (4 stages in total, 3 resblocks per stage). 4) Upsampling chain: mirror symmetry (4 upsampling, 3 resblocks per stage). 5) Channel table (write down the channels of each resolution to prevent readers from guessing "how many channels are 4x4"): - Let C = 128 - 64x64: 128 - 32x32: 256 - 16x16: 384 - 8x8: 512 - 4x4 (bottleneck): 512 6) Attention: Insert self-attention; 4 heads (the total number of channels remains the same) at both 16x16 and 8x8 resolutions (see 2.2). 7) Conditional injection: Use scale-and-shift form GroupNorm(h) * (w + 1) + b (see 2.3).
Why	
Common Issues	
Fix	1) Write the above 7 fields into your own config and print them out in each experiment output (to prevent misuse of old configurations). 2) Any "change in depth/change in width" must fall on changes to these 7 fields, and be aligned and evaluated under a fixed sampling protocol (see 5.6).

2.1 When U-Net is used as the backbone, the channel/resolution/residual block organization method Citation: P1 p11

How

1) First write the resolution chain as a definite value (do not leave it to the reader to guess): Assume the lowest resolution is 4x4, then the number of downsampling is $n_{down} = \log2(H / 4)$.
- Example: 64x64 -> 32 -> 16 -> 8 -> 4, so $n_{down} = 4$.
- Example: 32x32 -> 16 -> 8 -> 4, so $n_{down} = 3$.
- Example: 256x256 -> 128 -> 64 -> 32 -> 16 -> 8 -> 4, so $n_{down} = 6$.

2) According to the ImageNet-64 configuration of P1, a set of "default U-Net" that can be directly reproduced is given (write down each number):
- Resolution stages (high->low): 64, 32, 16, 8, 4
- Downsampling times: 4
- Residual blocks per downsampling stage: 3
- Upsampling stack: as a mirror of the downsampling stack (also 3 residual blocks per stage)
- Channel design (hard-coded according to resolution): Assume C = 128, then
- 64x64: 128

- 32x32: 256
- 16x16: 384
- 8x8: 512
- 4x4 (bottleneck): 512
- Attention insertion point and heads: see 2.2 (P1 is 16x16 and 8x8, 4 heads)
- Scale anchor (for reporting/reproduction): About 120M parameters in $C = 128$, about 39B FLOPs in a single forward

3) Then write "how much to take channel C" as discrete gears (don't leave it to the reader's subjective settings): P1 gives 4 reproducible gears in ImageNet-64, you only need to choose one of them to start:
- first-layer channels(=C): 64 / 96 / 128 / 192
- Number of parameters (given in the paper's figure): 30M / 68M / 120M / 270M
- Learning rate scaling rule (clearly written in P1): The 128-channel model $lr = 1e-4$ is used as the benchmark, and other models use $lr = 1e-4 / \sqrt{(C / 128)}$.
- Example: C=64 -> $lr \approx 1.41e-4$; C=96 -> $lr \approx 1.15e-4$; C=192 -> $lr \approx 8.16e-5$
- When writing experimental records, you are only allowed to write "one of C=64/96/128/192", and you are not allowed to write "C=160 (any trial)", which is an uncontrollable scale.

4) According to the CIFAR-10 configuration of P1, a set of "small models" that can be directly reproduced is given (the numbers are also hard-coded):
- Resolution stages (high->low): 32, 16, 8, 4
- Downsampling times: 3
- Residual blocks per downsampling stage: 3
- Channel design (hard-coded according to resolution): Assume $C = 128$, then
- 32x32: 128
- 16x16: 256
- 8x8: 256
- 4x4 (bottleneck): 256

5) Write "what you actually used" as a configuration table and fix it with the experiment: resolution chain, channel table, resblocks, attention resolution and heads, whether to use scale-shift conditional injection (see 2.3).

Why

Common Issues

Fix

1) Each time before expanding C or deepening the block, first write batch / lr / EMA together as a linkage super-parameter plan (see 4.2).
2) When evaluating, don't just look at the training loss: after fixing the sampler and sample size, look at the FID and NLL/likelihood side indicators at the same time (see 5.5/5.7).

2.2 Attention layer and multi-head setting (at which resolutions of U-Net are inserted) Citation: P1 p11

How	1) First, write "focus on only a few resolutions" as a configuration item (to avoid focusing on all resolutions and causing FLOPs to explode). 2) Press P1 settings to start by inserting attention into the two resolutions of 16x16 and 8x8. 3) The multi-head setup uses 4 heads as the starting point and keeps the total channel unchanged (P1). 4) Implement "4 heads" into an implementation definition (to avoid readers guessing): For an attention layer with a channel number of d_{model}, the channel of each head is $d_{head} = d_{model} / 4$, and the total dimension of the Q/K/V projection output is maintained as d_{model}.
Why	
Common Issues	
Fix	1) First perform 3 sets of ablation under a fixed sampling protocol: no attention / only 16x16 / 16x16+8x8, and confirm the benefits and FLOPs cost. 2) If the mixed accuracy is unstable: first fall back to FP32 to clearly verify "whether attention brings benefits"; only enable FP16 when throughput becomes a bottleneck, and use the same evaluation protocol to check that the quality does not degrade (see 4.3/5.6).

2.3 Time/noise condition injection method (Scale-Shift form) Citation: P1 p11

How	1) Prepare temporal/noise embedding vector emb(t) for each residual block. 2) Use MLP to predict the two parameters of w, b from emb(t), and hard-code the dimensions: if the number of channels of the block is c, then the output w, b are all vectors with a length of c (corresponding to one zoom and translation for each channel). 3) After normalization, do scale-and-shift: GroupNorm(h) * (w + 1) + b (the clear way of writing P1). Do not use the "add first and then normalize" way of writing such as GroupNorm(h + v).
Why	P1 pointed out that this modification has a slight improvement in the FID of ImageNet-64; the essence is to allow the conditions to modulate the normalized features in an affine form, which is more flexible.
Common Issues	
Fix	1) Write conditional injection formulas into configuration and logs (equally important as schedule/sampler). 2) If the reproduction fails, prioritize checking whether the injection point (before/after GN), the writing method of w + 1, and the embedding dimension/scaling of t are consistent.

2.4 Network preconditioning and noise embedding of EDM Citation: P3 p45-p46

How	1) Press P3 (EDM) to write the network into an explicit precondition form (write down the formula so that readers can reproduce it directly): - Let $\sigma_{data} = 0.5$ (default constant for P3) - $c_{skip}(\sigma) = \sigma_{data}^2 / (\sigma^2 + \sigma_{data}^2)$ - $c_{out}(\sigma) = \sigma * \sigma_{data} / \sqrt{(\sigma^2 + \sigma_{data}^2)}$ - $c_{in}(\sigma) = 1 / \sqrt{(\sigma^2 + \sigma_{data}^2)}$ - $c_{noise}(\sigma) = \log(\sigma) / 4$ (scalarized aperture of noise condition input) 2) Choose the noise embedding implementation and write it down: P3 compared DDPM positional and NCSN++ random Fourier features; you must explicitly choose one in the implementation, do not "arbitrarily set an embedding". 3) The supporting training terminal writes $p_{train}(\sigma)$ and $\lambda(\sigma)$ as configurations and enables them together with the preconditions (see 3.4). Otherwise, it is difficult for the preconditions to bring stable benefits alone.
Why	
Common Issues	
Fix	1) Bind precondition (2.4), $p_{train}(\sigma)/\lambda(\sigma)$ (3.4), and sampler (5.3-5.4) into the same configuration group and switch them on and off together. 2) After fixing the evaluation link (deterministic sampler + fixed NFE + fixed sample size), ablate it item by item to determine whether the gain/degradation is caused by the "training end" or the " sampling end".

2.5 How to “ redistribute ” structural capacity to reduce overfitting Citation: P3 p46

How	1) The total number of parameters/training budget is fixed, and only the "capacity distribution" is changed (to avoid misattributing benefits to increased computing power). 2) According to P3's rebalancing idea: reduce the number of layers/width at the minimum resolution (for example, 4x4), and move the capacity to a medium resolution (for example, 16x16). 3) Align and compare using a fixed sampling protocol to observe whether overfitting/duplicate samples are less likely to occur.
Why	This experience comes directly from P3's config ablation, indicating that "deeper/lower resolution" is not necessarily better, and that capacity should be placed in more critical resolution segments.
Common Issues	If you blindly increase the number of lowest resolution layers, the training loss may be lower but the FID/visual quality will become worse (overfitting or sample diversity decreases).
Fix	1) The total number of parameters/training budget is fixed, and only the "capacity distribution" is changed (4x4 vs 16x16, etc.). 2) Align the evaluation with a fixed sampler and a fixed sample size, and draw the FID curve and sample grid during the training process. 3) If overfitting is lighter or the quality is more stable, use this capacity distribution as the new default and continue to adjust and optimize the hyperparameters (see 4.2).

2.6 "Dynamic calculation" (DyDiT) of Transformer routes Citation: P5 p1,p5-p6

How	1) Clear dynamic granularity (P5 provides two main lines): (a) timestep-wise: different t uses different calculations; (b) token-wise: allows bypass calculations for some tokens (such as MLP bypass). 2) Let ’ s start with token-wise MLP bypass (preserving attention interaction), and connect the router/dynamic mask to the backbone as an additional module. 3) Use STE/Gumbel-Sigmoid to train the router, and introduce FLOPs constraints to write the budget into the target (see 3.5).
Why	P5 observes that the difficulty of noise prediction is uneven at different time steps and different spatial patches; dynamization focuses computing power on "harder steps/harder tokens" to improve/maintain quality at the same or lower FLOPs.
Common Issues	
Fix	1) First let the dynamic structure "only rely on timestep": precompute mask/activation to avoid sample-dependent graph causing a decrease in inference throughput (P5). 2) If training is unstable: Enable warm-up, keep $L_{complete}$ for a period of time and then remove it (see 4.4). 3) Use FLOPs constraints to verify whether the budget is up to standard, and then check whether the quality is maintained (see 3.5).

2.6* "Depth/number of heads/width" configuration table of Transformer backbone (static DiT) (can be reproduced directly) Citation: P5 p17 (Table 7)

How	This table specifically answers "How much depth and heads to choose for Transformer", and readers do not need to read the paper. 1) Write the three core structural hyperparameters of Transformer as explicit configuration items: - layers(depth): block number - heads: Number of long positions - channel(width, can also be written as d_{model}): channel dimension of token 2) Directly select a discrete level from Table 7 of P5 as the starting point of the "static DiT trunk" (do not create your own depth/number of heads): - model | params(M) | layers | heads | channel | pre-training source - DiT-S | 33 | 12 | 6 | 384 | 5M iter (Pan et al., 2024) - DiT-B | 130 | 12 | 12 | 768 | 1.6M iter (Pan et al., 2024) - DiT-XL| 675 | 28 | 16 | 1152 | 7M iter (Peebles & Xie, 2023) 3) To make DyDiT: first run through the static DiT (training-sampling-evaluation closed loop), and then add a router to make it dynamic; the P5 table points out that DyDiT "layers/heads/channels remain unchanged" at these levels, but only introduces a controllable activation mask and a small number of additional parameters.
Why	
Common Issues	
Fix	1) First fix a gear (it is recommended to start with DiT-S: 12 layers, 6 heads, width=384), and use the fixed sampling protocol to run through the visualization and evaluation closed loop (see 5.0/5.5/5.6). 2) The upgrade scale only allows jumping gears: S -> B -> XL; and each upgrade must also record whether the training hyperparameters (optimizer/lr/batch/warmup) are linked according to the caliber of the paper (Table 6 of P5 p17; see Appendix A.2).

2.7 Covariance/variance modeling as a structural extension of " accelerated sampling " Citation: P1 p2-p3, P6 p1-p4

How	1) First write down your goal clearly: is it "few-step (NFE 5/10/20) quality" or "high NLL/likelihood" or both. 2) If the goal contains few steps or likelihood: enter Σ_θ into the design list as a structural extension rather than a fixed heuristic. 3) Choose one or a combination of two routes within the thesis: - Route P1: Use L_{hybrid} to learn variance (see 2.8/3.2/3.3). - Route P6: Predict optimal diagonal covariance with OCM amortization (see 3.7/5.8).
Why	P6 clearly points out that covariance selection has a huge impact when sampling with few steps; better diagonal covariance can improve quality, recall and likelihood with few NFE.
Common Issues	
Fix	1) If your bottleneck is low-step quality: Prioritize changing the sampling process (DDIM/EDM/Heun) to confirm that "the link can be accelerated" (see 5.1-5.4). 2) If it is still poor: introduce variance/covariance learning again. - Learning variance: L_{hybrid} + bounded interpolation (see 3.2/2.8). - Learning diagonal covariance: OCM training covariance header (see 3.7/5.8).

2.8 Parameterization of learning variance: interpolating between upper and lower bounds Citation: P1 p3-p4

How	1) Define upper and lower bounds: $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \tilde{\beta}$ (P1 regards them as feasible boundaries). 2) Let the network output the interpolation coefficient v, and constrain the variance between the two in log-variance space (maintaining positive definite and numerical stability). 3) Enable L_{hybrid} training variance (otherwise L_{simple} has no learning signal for variance, see 3.2).
Why	
Common Issues	The learned variance is too large, resulting in excessive noise in short trajectories; the learned variance is too small, resulting in reduced diversity/coverage or unstable sampling.
Fix	1) Confirm that the variance header has a learning signal: L_{hybrid} must be enabled and not just L_{simple} (see 3.2). 2) During training, draw a "less-step-quality curve": 10/20/50/100 steps (DDIM/deterministic) as a verification panel (see 5.1/5.5). 3) If there is excessive noise/decreased diversity: Return to the interpolation range and λ weight, first reduce the VLB weight and then review.

2.9 Noise scheduling (Linear vs Cosine) is the core knob of training and sampling Citation: P1 p4-p5

How	1) Implement two sets of scheduling, Linear and Cosine, and switch them to ensure consistent calculation of $\bar{\alpha}_t$. 2) Schedule comparison under the same structure/same sampling protocol: report FID and bits/dim(NLL) at the same time. 3) If using Cosine: Adjust dropout/training duration as a linkage hyperparameter (see 4.5), and use a denser snapshot curve to check whether it enters the overfitting interval faster (see 5.6).
Why	
Common Issues	
Fix	1) Schedule changes must be linked to regular rules: do dropout small grids under Cosine, do not just change the schedule (see 4.5). 2) Use snapshot + deterministic sampler evaluation curve to locate the overfitting window: when " loss decreases but FID increases", priority is given to regularization/early stopping rather than continuing to accumulate computing power (see 5.6).

3) Loss function selection

3.0 Getting started with this chapter: Use `L_hybrid` to learn the minimum implementation of variance (output header + loss calculation) Citation: P1 p3-p4

How	Think of it as an "implementation checklist", from model output to loss. 1) The model output contains at least two channels: - Noise prediction $\varepsilon_\theta(x_t, t)$ (for L_{simple}) - Variance interpolation coefficient $v_\theta(x_t, t)$ (used to construct Σ_θ, see 2.8) 2) Construct the diagonal variance Σ_θ using "bounded interpolation" of 2.8 (constraining the variance between β_t and $\tilde{\beta}$). 3) Compute two losses and weight them: - $L_{simple} = E \| \varepsilon - \varepsilon_\theta \|^2$ - L_{vlb} (term corresponding to variational lower bound, whose gradient is used to train the variance branch) - $L_{hybrid} = L_{simple} + \lambda * L_{vlb}$, and start with $\lambda = 0.001$(P1). 4) The log must be split and recorded: L_{simple}, L_{vlb}, L_{hybrid} three curves; otherwise you cannot judge whether "the backbone is broken" or "the variance term is shaking".
Why	P1 clearly states that L_{simple} does not give variance learning signals; L_{hybrid} adds VLB back with a small weight, so that variance can be learned without significantly harming sample quality.
Common Issues	Only the variance head is turned on but still only trains L_{simple}, causing the variance head to never converge; or λ is too large to cause VLB jitter to overwhelm training.
Fix	1) If L_{vlb} has large jitter: first enable time step resampling/importance sampling to stabilize it (see 3.3). 2) If the sample quality decreases: first lower λ by one level and recheck; do not change the network structure first.

3.1 The relationship between `L_simple` and VLB (and when to use whom) Citation: P1 p2-p3

How	1) By default, $L_{simple} = E \| \varepsilon - \varepsilon_\theta \|^2$ is used to run through the "sample quality baseline" (minimum variables). 2) Specify whether your additional goals include: learning variance/covariance, or focusing on log-likelihood/NLL. VLB dependencies are added only if these targets are present. 3) Write the training target as a switchable configuration: loss $= L_{simple}$ or loss $= L_{hybrid}$, and compare it under a fixed sampling protocol (see Chapter 5).
Why	P1 Summary: Optimizing VLB directly tends to have poor sample quality; L_{simple} is better in practice as a reweighted form, but it does not train variance.
Common Issues	Focusing only on L_{simple} can make "low loss but still poor sampling" situations difficult to interpret, especially if the sampler or variance settings do not match.
Fix	1) When the sampling quality is poor but L_{simple} is already low: Prioritize checking whether the sampling link (DDIM/Heun/randomness) matches the training settings (see 5.1-5.4). 2) When the target contains variance/likelihood: use L_{hybrid} + time step resampling to stabilize VLB (see 3.2-3.3); or directly take the OCM route to change the covariance (see 3.7).

3.2 `L_hybrid = L_simple + lambda * L_vlb` is used to learn variance Citation: P1 p3

How	1) Implement L_{vlb} and record its value separately in the log (do not mix it with L_{simple}). 2) Enable mixed target: $L_{hybrid} = L_{simple} + \lambda * L_{vlb}$, starting from the starting point of P1 $\lambda = 0.001$. 3) Matching variance parameterization: Use "bounded interpolation" of 2.8 to keep Σ_θ always within a reasonable range.
Why	
Common Issues	
Fix	1) If training jitter: First confirm that λ is small enough and L_{vlb} does not have numerical explosion; then enable time step resampling (see 3.3). 2) If the variance header does not converge: Make sure you are indeed training L_{hybrid} (only L_{simple} will not give a variance signal).

3.3 Use importance sampling/resampling to reduce the variance of VLB terms Citation: P1 p4

How	1) When L_{vlb} / L_{hybrid} is enabled, changes the Time Step Sampling Distribution from uniform to an estimable/resampleable distribution. 2) Count the VLB terms (mean/variance or proxy) of each t, and assign higher sampling probability to high variance/high impact time steps accordingly. 3) Write this sampling strategy into the configuration and fix it, otherwise the VLB jitter between different experiments will not be comparable.
Why	P1 observes that the contribution of each time step of VLB varies greatly, and uniform sampling will introduce unnecessary noise; importance sampling can significantly reduce the target variance and improve optimization.
Common Issues	The VLB curve is very jittery, and the training seems to "not converge" or requires a smaller learning rate; the effect is unstable under the same computing power.
Fix	1) First, connect the timestep resampling switch to the training pipeline, and write the "estimated diameter of the resampling distribution" into the configuration (to avoid inconsistency in each training). 2) If the VLB is still shaking: Reduce λ, increase the batch size, or delay enabling the VLB item (use L_{simple} to warm up first) until the backbone is stable before turning it on.

3.4 EDM: Noise distribution `p_train(sigma)` and loss weighting `lambda(sigma)` Citation: P3 p45

How	Directly copyable "Configuration Groups". 1) Training noise distribution (write down the default constants so that readers can reproduce them directly): - $\ln(\sigma) \sim N(P_{mean}, P_{std}^2)$, the default for P3 is $P_{mean} = -1.2$, $P_{std} = 1.2$ - $\sigma_{min} = 0.002$, $\sigma_{max} = 80$ - $\sigma_{data} = 0.5$ 2) loss weighting (write down the formula): - $\lambda(\sigma) = (\sigma^2 + \sigma_{data}^2) / (\sigma * \sigma_{data})^2$ 3) Linked with preconditions: $p_{train}(\sigma)$, $\lambda(\sigma)$, and preconditions (2.4) must be enabled and ablated together; do not change just one of the knobs.
Why	P3 emphasizes that the "denoising difficulty/error structure" of different sigma is different; unreasonable distribution or weight will cause the model to waste capacity in invalid intervals or ignore key intervals.
Common Issues	Generated results exhibit systematic color cast/oversaturation drift; or are only good at certain noise levels, but the overall sampling link is unstable.
Fix	1) The fixed network and optimized hyperparameters remain unchanged. First, only the configuration groups of $p_{train}(\sigma)$ and $\lambda(\sigma)$ are scanned to avoid misattributing the benefits to other places. 2) Each configuration uses a deterministic sampler + fixed NFE to evaluate FID/NLL, and saves a fixed x_T sample grid for quality inspection (see 5.2/5.6).

3.5 DyDiT: FLOPs combined with constrained loss and diffusion primary loss Citation: P5 p5-p6

How	1) Define static baseline FLOPs: measure F_{static} (fixed structure once forward). 2) Statistical dynamic FLOPs: $F_{dynamic}$ is counted by batch during training (the FLOPs of the actual execution subgraph). 3) Set target budget ratio λ : How close you want avg($F_{dynamic}$ / F_{static}) to be. 4) Joint training: $L = L_{DiT} + (avg(F_{dynamic} / F_{static}) - \lambda)^2$, and separately record in the log whether the budget items meet the standards.
Why	Dynamic routing will naturally tend to "increase more computing power to reduce losses"; the FLOPs constraint explicitly binds the training goal to the inference budget, making the efficiency-quality tradeoff controllable.
Common Issues	Using only diffusion loss to train dynamic structures will lead to uncontrollable inference FLOPs; or sacrificing key modules to reduce FLOPs (quality collapse).
Fix	1) If the budget is up to standard but the quality is poor: Prioritize reducing the dynamic range (first only do MLP token bypass and retain MHSA interaction) (P5). 2) If the quality is up to standard but the budget is not up to standard: Adjust the target ratio λ or increase the weight of FLOPs constraints, and check whether the caliber of statistics $F_{dynamic}$ is consistent.

3.6 "Poor quality of few-step sampling" is often a problem of variance/covariance selection Citation: P6 p1-p3

How	1) Write the less-step goal as the default evaluation panel: 5/10/20 steps (NFE) three-speed fixed comparison. 2) Compare covariance strategies under this panel: heuristic vs learned variance (P1) vs OCM (P6). 3) Record FID and NLL/recall at the same time to avoid misjudgment by just looking at a single indicator (see 5.5/5.8).
Why	
Common Issues	Using heuristic variance the FID ranges poorly or generates divergence at 5/10/20 steps; but adding the number of steps back "looks good".
Fix	1) First change to a more stable process with fewer steps: DDIM (eta=0) or EDM/Heun, and stabilize the fundamentals of "still making a picture with fewer steps" (see 5.1/5.3). 2) Introduce covariance learning: use OCM or learning variance to improve the quality/likelihood of few steps (see 3.2/3.7).

3.7 OCM: Optimal diagonal covariance (diagonal Hessian) regression with unbiased objective Citation: P6 p3-p4

How	1) Clarify the cost of direct estimation: P6 points out that M> = 100 Rademacher samples are needed to obtain ideal diagonal approximation on CIFAR-10; and each sample requires one forward + one back pass, the total cost is about 2M network evaluations, and inference is not available. 2) Add a diagonal Hessian/covariance prediction header $h_\phi(\tilde{x})$, in parallel with the score/mean prediction. 3) Train the head with the unbiased OCM objective: use v~Rademacher to construct the Monte Carlo term of $v \odot H(\tilde{x}) v$ and let h_ϕ match its expectation (amortized estimate). 4) Use the h_ϕ output to construct the diagonal covariance during sampling, and evaluate whether FID/NLL is improved under the 5/10/20 step panel (see 5.8).
Why	The goal is to avoid the bias caused by "small M direct regression to the mean", and at the same time, only one network forward can be used to obtain the diagonal term during inference.
Common Issues	
Fix	1) Training period: First train and stabilize the OCM head (monitor its diagonal term error or related proxy indicators), and then connect it to the sampling link. 2) Evaluation period: Fixed the few-step panel at 5/10/20 steps, and reported FID and NLL at the same time. Only stable improvement in the few-step regime can be considered as "returning to the training complexity" (see 5.8).

4) Optimizer selection

4.0 Getting Started Example in this Chapter: Adam + EMA ’ s “ Minimum Training Protocol ” (What You Should Fix) Citation: P1 p11; P3 p45

How	
Why	P1/P3 both regard EMA as the default training component; not fixing "whether to use EMA" will lead to incomparable sampling quality under the same loss.
Common Issues	
Fix	1) First unify the EMA weights for evaluation, and then discuss whether it is necessary to adjust the EMA strength (see the grid in 4.1). 2) Write the EMA configuration into config and log, and do not allow "memory" reproduction.

4.1 Adam + EMA is the default starting point in the paper system. Citation: P1 p11; P3 p45

How	1) The optimizer uses Adam as the default starting point. 2) Maintain the EMA weight from the first step of training, and clarify the "EMA weight for evaluation/sampling" (P1/P3 both regard EMA as the default component). 3) Start running with the starting point hyperparameter of P1: lr $=$ $1e-4$, batch $=$ 128, EMA $=$ 0.9999; if expressed according to P3, write EMA as half-life (Mimg) and record it. 4) Comparison in the log: Original weight vs EMA weight results under the same sampling protocol to avoid misjudgment of whether EMA is valid.
Why	EMA can significantly smooth training noise and improve sampling stability; both papers regard EMA as a key training component and write it into the default hyperparameter table.
Common Issues	
Fix	1) After fixing the sampling protocol, make a small grid for EMA: change only one dimension of EMA value/half-life, and observe the changes in sampling stability and FID/NLL. 2) In any report, both "training weight" and "sampling EMA weight" should be written to avoid using the wrong weight when reproducing the experiment.

4.2 Linkage between learning rate/batch and model size (hyperparameters should not be regarded as constants) Citation: P1 p11; P3 p45

How	Give two sets of default starting points that are directly reproducible. 1) Default starting point A (P1, used as baseline for most experiments): Adam + EMA, and fixed: - batch=128 - lr=1e-4 - EMA=0.9999 2) Throughput-oriented starting point B (P3, CIFAR-10 ' s " Ours " training table): Directly reproduce its training protocol (write down each number, no need to read the paper): - Total training volume: 200 Mimg - Number of GPUs: 8 - batch: 512 - Learning rate: 1e-3 - LR ramp-up: 10 Mimg - EMA half-life: 0.5 Mimg - Gradient clipping: Off (P3 experience is that turning it off has no effect and is more trouble-free) - Dropout: 13% (P3 gives its final choice in this table) - Data set x-flips: closed (with non-leaky augmentation idea) - Augment probability: 12% 3) When changing the model size (for example, making C larger) or changing the degree of parallelism, "changing only the model without changing the hyperparameters" is not allowed: the comparison (A or B) must be re-run at least once before continuing to fine-tune.
Why	The gradient noise of diffusion training is obviously coupled with the effective batch and target item variance; fixed lr often leads to slow or unstable convergence.
Common Issues	After increasing the model, the loss decreases slowly, oscillates, or training diverges; comparison between different configurations is distorted.
Fix	1) After expanding the model, first do a comparison of "only changing the optimized hyperparameters": fix the model and sampling protocol, and scan the batch/lr/warmup/EMA time constant. 2) If VLB/covariance learning is enabled: Also enable stabilization strategies such as time step resampling or OCM, otherwise training jitter will mask structural gains (see 3.3/3.7).

4.3 Gradient clipping and mixing accuracy: subject to "whether there is benefit" Citation: P3 p45-p46

How	1) Gradient clipping: Strictly follow the conclusion of P3 and implement it as "off by default": - Write the switches into config (eg grad_clip_norm: null / grad_clip_norm: 1.0). - Fix everything else (model/loss/sampler/NFE/sample size) and do only one on/off comparison; if there is no stable gain, just use P3's default: off. 2) Mixed precision: Implemented strictly according to the usage scenarios of P3: - Only enable FP16 when throughput is required for large-scale training (such as P3's ImageNet-64 configuration). - The log must indicate: whether it is FP16, whether there is loss scaling, and whether NaN/Inf occurs (if it occurs, it will fall back to FP32 and run through the baseline first). 3) Use the same evaluation protocol for any "stabilization means" to confirm that "it really improves the sampling quality/stability", don't just look at loss (see 5.6 Deterministic Sampling Point Selection Protocol).
Why	
Common Issues	Blind cropping causes the effective learning rate to be lowered; FP16 suffers from numerical instability in some layers, resulting in training collapse or quality degradation.
Fix	1) First, remove the cropping switch in the baseline configuration, leaving only the settings that " really improve stability/quality". 2) If NaN/divergence occurs under mixed precision: first fall back to FP32 to run through the training-sampling closed loop; then re-enable FP16 only as a throughput optimization according to the P3 idea, and use the same evaluation protocol again to verify that the quality does not degrade.

4.4 End-to-end training stabilization of dynamic structures Citation: P5 p6

How	1) The router uses differentiable approximations such as STE/Gumbel-Sigmoid to train discrete decisions (P5). 2) If the end-to-end finetune is unstable: introduce warm-up first, and retain the complete model supervision item $L_{complete}$ in the early stage of training. 3) When the router no longer collapses and the budget items meet the standards, $L_{complete}$ will be gradually removed and pure joint target training will be entered (linked with 3.5).
Why	
Common Issues	Unstable oscillations during training, router collapse (fully on or off), and significant degradation in quality.
Fix	1) Use warm-up to stabilize the training first, and then gradually remove $L_{complete}$; do not let the router completely take over at the beginning (P5). 2) Use FLOPs constraints to ensure that the inference budget meets the standards, and review quality without systematic drops under a fixed sampling protocol (see 3.5/5.6).

4.5 Dropout and EMA need to make small grids according to the schedule/data set. Citation: P1 p11; P3 p45

How	1) Dropout's "starting grid" directly reproduces P1's CIFAR-10 grid: {0.1, 0.2, 0.3} (only dropout is changed each time, the rest remains unchanged). 2) Dropout's "single starting point" can also directly reproduce the final values of P3 Table 7 (when the budget is insufficient, use them to run through the closed loop first): - CIFAR-10: 13% (improved configuration of P3) - FFHQ: 5%, AFHQv2: 25%(P3) - ImageNet-64: 10%(P3) 3) EMA's "single starting point" also directly reproduces the half-life of P3 Table 7 (using half-life is less likely to be confused with batch/lr): - CIFAR-10/FFHQ/AFHQv2 (improved configuration): 0.5 Mimg - ImageNet-64: 50 Mimg 4) Make a dropout grid for each noise schedule (Linear/Cosine) (P1 pointed out that "the optimal dropout changes with the schedule"). 5) The dropout/EMA comparison must be done under a fixed sampling protocol (deterministic sampler + fixed NFE + fixed sample size) (see 5.6).
Why	The effective regularization strength of diffusion training is coupled with noise scheduling, enhancement, and sampler selection; dropout and EMA determine the "stability-fitness" balance point.
Common Issues	Too small a dropout leads to overfitting or reduced sample diversity; too large a dropout leads to underfitting and blurred details; Improper EMA setting causes the sampling weight to lag behind or fluctuate greatly.
Fix	1) Incorporate dropout and EMA into the small grid at the same time, but only change a small number of combinations each time to control the experimental cost. 2) The selection criteria are mainly FID/NLL/sample diversity under a fixed sampling protocol, and don ’ t just look at the training loss.

5) Visualization and fine-tuning (incorporating the “ sampling link ” into the parameter-tuning closed loop)

5.0 Getting started with this chapter: Generate comparable visualizations for each epoch (fixed `x_T` + fixed sampler) Citation: P2 p7-p8; P3 p45

How	Turn "visualization" into a reproducible evaluation protocol rather than randomly sampling a small number of images. 1) Fix a set of initial latent x_T (for example, save a noise tensor with shape [64, 3, H, W], or fix a random seed). 2) Fixed sampler as visualization baseline: - Few steps: use DDIM, $\eta = 0$ (deterministic) - Fixed number of steps: This manual is fixed at $N = 50$ by default (unchanged throughout; if you want to make it more stable, you can add a $N = 100$ control panel) 3) At the end of each epoch, use "the same x_T" to generate the same 8x8 grid map and save it, so that the changes you see only come from model weight changes (P2's consistency diagnosis idea). 4) If adjusting the numerical integration of EDM: fix the solver to Heun + fixed NFE, otherwise the visualization will be contaminated by sampling noise (P3).
Why	
Common Issues	
Fix	1) Forcefully fix x_T with the sampler, writing it into the code as default. 2) To show variety, save a set of "multi-seed grids", but do not replace the main panel with fixed x_T.

5.1 DDIM: Keep training unchanged, only change the sampling to obtain high quality in few steps Citation: P2 p6-p8

How	1) Keep training unchanged: $T = 1000$ remains unchanged from the training target. 2) Change the sampling to DDIM: select the subsequence τ and choose one of the two rules in P2 (don't invent the third one yourself, reproduce the paper first): - Linear: $\tau_i = \text{floor}(c * i)$, choose c to make tau_{N − 1} close to T. - Quadratic: $\tau_i = \text{floor}(c * i^2)$, choose c so that tau_{N − 1} is close to T. - Empirical selection of P2: quadratic for CIFAR-10 and linear for the rest of the data sets. - Fixed number of steps sweep (default panel): $N \in \{10, 20, 50, 100\}$. 3) First fix $\eta = 0$ as a deterministic baseline; then try $\eta > 0$ to see if it improves diversity. 4) Implement each step according to the closed update formula of P2 (see the formula in 0.3); after implementation, first use $\eta = 0$ to review "the high-level semantic consistency of different N under the same x_T" (see 5.2). 5) Each step number is compared under the same evaluation protocol: fixed sampler, fixed sample size, fixed whether to use EMA weight (see 5.5-5.6).
Why	P2 proves that DDIM is often significantly better than DDPM with fewer sampling steps; and points out that DDIM can also approach 1000 step quality on shorter trajectories, bringing 10x-50x acceleration.
Common Issues	
Fix	1) First use $\eta = 0$ to run through the few-step baseline (50/100 steps) to confirm that the implementation is correct and the curve is monotonic and reasonable. 2) Then scan η under fixed x_T and fixed evaluation protocol to observe whether there is a stable tradeoff between diversity and quality (see 5.2/5.5).

5.2 Consistency diagnosis of DDIM: fix the same `x_T` and see different trajectories Citation: P2 p7-p8

How	1) Fixed a set of initial latent x_T (fixed random seed or cached tensor directly). 2) Use different τ (different step number/time point distribution) to sample the same x_T. 3) Do a consistency check: the high-level semantics (subject/composition) should be basically consistent, and the details only change with the number of steps. 4) If the consistency is poor: treat it as a red flag of sampling implementation/numeric error/scheduling mismatch, check the implementation first and then adjust parameters.
Why	P2 observed that under fixed x_T, the high-level features generated by different trajectories of DDIM are similar, implying that x_T is an informative latent code; this provides a "structural signal" for parameter adjustment.
Common Issues	
Fix	1) First switch to $\eta = 0$ and use the smoother/dense τ subsequence to eliminate the effects of randomness and too coarse step size. 2) If it is still inconsistent: Prioritize checking whether the $\bar{\alpha}_t$ value, mean/variance term implementation, and broadcast dimensions are correct (this is one of the most common causes of " looks like noise").

5.3 EDM: 2nd order ODE solver (Heun) is often more cost-effective than Euler Citation: P3 p45

How	1) Abstract the sampler implementation into "a numerical integrator for a given noise sequence $\{\sigma_i\}$", and hard-code the default time step discretization (EDM formula) according to P3: - Assume the number of sampling steps is N, and P3 uses the σ sequence of ρ -warp: $\sigma_i = (\sigma_{max}^{(1/\rho)} + i/(N-1) * (\sigma_{min}^{(1/\rho)} - \sigma_{max}^{(1/\rho)}))^{\rho}$, where $i = 0..N-1$, and let $\sigma_N = 0$. - The default boundary selection of P3: $\sigma_{min} = \max(\sigma_{lo}, 0.002)$, $\sigma_{max} = \min(\sigma_{hi}, 80)$; the commonly used default constant is $\sigma_{min} = 0.002$, $\sigma_{max} = 80$. 2) Replace Euler with Heun (2nd order, ladder method) at the same number of steps N. - The cost caliber is hard-coded: P3 points out that Heun has one more D_θ evaluation per step; the total NFE when using Heun is approximately $2N-1$ (no secondary correction is required from the last step to $\sigma = 0$). 3) Only change the solver without changing other settings, and draw NFE-quality curve comparison under the fixed evaluation protocol.
Why	
Common Issues	Euler has large local errors in certain noise segments, which require many steps to compensate; deviation from the trajectory in few steps causes the image to look more like noise or have broken details.
Fix	1) Fix Heun as the default deterministic sampler, and first replace Euler to verify the returns under the same time point sequence. 2) If the benefits are limited: Then adjust the discrete time point/step distribution and use fixed NFE alignment comparison (according to the ablation method of P3).

5.4 EDM: Randomness is not “ the more the better ”, it needs to be empirically searched according to the data set. Citation: P3 p45

How

Directly available meshes come from P3's randomly sampled ablation table.
1) Write the randomness as a parameter group: S_{churn}, S_{tmin}, S_{tmax}, S_{noise}, and make sure it can be grid searched.
2) Deterministic sampling as baseline: $S_{churn} = 0$ (P3 is considered a deterministic sampler baseline).
3) Use the grid set of P3 as the starting point:
- S_{churn}: 0,10,20,...,100
- S_{tmin}: 0,0.005,0.01,0.02,...,1,2,5,10
- S_{tmax}: 0.2,0.5,1,2,...,10,20,50,80
- S_{noise}: 1.000,1.001,...,1.010
4) If you need a "single starting point" for fine-tuning: directly reproduce the used parameters of P3 Table 5 (you can then perform local searches within the grid):
- CIFAR-10 (VP/VE): $S_{churn} = 30$, $S_{tmin} = 0.01$, $S_{tmax} = 1$, $S_{noise} = 1.007$
- ImageNet-64 (P3 settings for pre-trained/ours): $S_{tmin} = 0.05$, $S_{tmax} = 50$, $S_{noise} = 1.003$; S_{churn} takes 80 or 40 in different settings (subject to the reproduction target table, but this must be written into the experiment log).
5) Fix the NFE and then search for randomness (to avoid attribution confusion): For the same N (or the same NFE), only change the randomness parameters and choose the best point; it is not allowed to change the time point discretization/solver/randomness at the same time.

Why

P3 points out that randomness helps correct sampling errors, but it can also introduce image degradation (such as oversaturation drift); the optimal randomness depends on the data set and training settings and must be determined empirically.

Common Issues

Too much randomness leads to color drift/oversaturation, and detail jitter; or randomness can actually harm FID under certain training settings.

Fix

1) First fix $S_{churn} = 0$ as the alignment baseline, and then turn on the randomness knobs one by one (to avoid changing too much at once without seeing the cause and effect).
2) If oversaturation/drift occurs: Prioritize limiting the effective range of S_{tmin} / S_{tmax} to avoid introducing randomness in the extremely low noise segment (heuristic motivation of P3).

5.5 Evaluation indicators and sample size: Trade-off between FID/NLL/Precision-Recall Citation: P1 p1,p11; P6 p3; P4 p37

How

1) Write the "evaluation sample size + sampler + NFE" as a protocol and fix it (otherwise FID/NLL will not be comparable between different experiments).
2) Write down the calculation caliber of FID (avoid "same name but different caliber"):
- Sample size: 50,000 generated images vs all real images (P3 specifies its default size in implementation details; P1 also uses 50K as a common benchmark).
- No enhancement is done on the real image side: do not use x − flips and wait for enhancement before calculating FID (P3 clearly states).
- To reduce random fluctuations: P3 reports that its FID will typically be recalculated 3 times and report the minimum value, noting that the random fluctuations are on the order of ±2%.
- Downsampling during fast ablation: P1 will use 10K samples to accelerate in the large ablation of unconditional ImageNet-64 (admittedly it will bring bias but used for relative comparison); if you use 10K, you must write it out explicitly in the figure/table notes and use 50K to review the key conclusions.
3) Fix the indicator panel to multiple indicators: FID (quality), NLL/bits-dim (likelihood), improved precision/recall (coverage) and record them simultaneously to avoid single indicator misjudgment (P1/P4/P6).

Why

P1 pointed out that looking at FID alone is not enough to judge coverage, precision/recall can reveal the high recall characteristics of the diffusion model; P4 also emphasized that the evaluation benchmark is still incomplete and susceptible to bias.

Common Issues

Fix

1) Establish a "two-level evaluation": fast ablation with fewer samples but fixed sampler and fixed NFE; key conclusions are reviewed with 50K samples (P1).
2) Only one degree of freedom change is allowed at a time (one of the three models/losses/samplers), otherwise FID changes cannot be attributed.

5.6 “ Save snapshots + choose the best with deterministic sampler ” as reporting protocol Citation: P3 p45

How	1) Write down the checkpoint frequency and "select the best" rules (write them into the experimental protocol, do not set them subjectively temporarily). 2) According to P3 method: save snapshot according to the number of training images (for example, every 2.5 Mimg). 3) Use a fixed deterministic sampler and fixed NFE to evaluate FID, and take the snapshot report with the lowest FID: - The implementation details of P3 are clear: its reporting is usually based on "deterministic sampler + fixed NFE" (such as NFE=35 or NFE=79, depending on the resolution) to select the best point. - You don't have to use the same NFE as it, but it must be fixed and documented in logs/reports. 4) Write out simultaneously when reporting: sampler (deterministic/random), NFE/number of steps, sample size, and selection rules (lowest FID/other indicators).
Why	P3 uses this protocol to reduce the contamination of conclusions by "evaluation randomness and sampler differences", making the comparison of the training process more like a reproducible numerical experiment rather than a sampling display.
Common Issues	
Fix	1) Use deterministic samplers to lock the evaluation link to avoid "point selection noise" caused by random samplers. 2) Checkpoint selection relies on the FID/quality curve under fixed NFE, rather than a single training loss (P3).

5.7 Few-step inference acceleration: covariance and few-step process should be modified together. Citation: P6 p1-p4; P2 p6-p8

How	1) Set a few-step goal first: 5/10/20 Step 3 is fixed as the default evaluation panel. 2) On the process side, change the sampling first: DDIM (deterministic priority) or EDM/Heun. After reducing the number of steps, the graph can still be stabilized (see 5.1/5.3). 3) Change the covariance on the statistical side: If the few-step quality/likelihood is still poor, introduce learning variance (P1) or OCM (P6) (see 3.2/3.7). 4) The indicator side must be linked: report FID and NLL/recall at the same time to avoid only looking at FID and ignoring coverage (see 5.5/5.8).
Why	
Common Issues	Only do DDIM but still use heuristics for variance/covariance, and the quality of few steps is still poor; or only learn covariance but the sampling process is still not suitable, and the benefits are limited.
Fix	1) Write the "less step goal" as the system goal into the experimental protocol: review the same set of 5/10/20 step panels repeatedly. 2) Press P2/P3/P6 to perform linked ablation on the sampling trajectory (DDIM), numerical integration (Heun/time point), and covariance (OCM/learning variance) respectively, and use the unified visualization panel to record the "number of steps-quality curve" and "coverage/likelihood curve".

5.8 The visualization goal of OCM: Directly view the (number of steps, FID, NLL) triples in less-step scenarios Citation: P6 p3

How	1) Fix the same data set and model (keep the weights unchanged), and only change the covariance strategy (heuristic vs OCM). 2) Fixed few-step set: 5/10/15/20 steps (or your target NFE set). 3) Report FID and NLL simultaneously at each step, and attach the same batch of fixed x_T sample grids (see 5.2). 4) Use the "(number of steps, FID, NLL) triplet curve" to determine the true benefit of OCM in a few-step regime (P6).
Why	The claim of P6 is that "covariance is critical for few-steps"; therefore the visualization must fall directly on the few-step regime, otherwise the core benefit of covariance improvement will be obscured.
Common Issues	Only evaluating at 1000 steps, the value of covariance learning cannot be seen; or only looking at FID cannot see changes in likelihood/coverage.
Fix	1) Fix the few-step evaluation panel to the default: scan the number of steps and record FID and NLL at the same time. 2) Only when OCM stably brings benefits on the few-step panel will the covariance head and its training complexity be included in the long-term default configuration (P6).

Appendix) Quick table lookup (only the tables and figures in P1-P6 that “ can be reproduced directly ” are included)

A.0 "Indicator → Model" decision-making process for three types of data sets (image/video/SMPL-H) Citation: P4 p37; P1 p11; P3 p45; P5 p17; P6 p3

How

Use a 5-step pipeline to map data indicators to model size (engineering rule, not a hard threshold from papers).
1) Collect four indicator groups (shared by image/video/SMPL-H):
- Scale: N_{eff} (deduplicated effective sample count).
- Quality: bad_{rate}, dup_{rate}, $align_{err}$.
- Complexity: image diversity, video motion/long-range dependency, SMPL-H dynamics/hand details.
- Alignment: all text-conditioned tasks must report $align_{err}$.
2) Compute scores and gate:
- $S_N = clip((log10(N_{eff}) - 4) / 2, 0, 1)$
- $S_Q = 1 - clip((bad_{rate} + dup_{rate} + align_{err}) / 0.45, 0, 1)$
- S_C by modality: image=normalized diversity; video=$0.5 * motion + 0.5 * long_{range}$; SMPL-H=$0.5 * dyn + 0.5 * hand_{detail}$
- usability gate G: if dup_{rate}>0.20 or $align_{err}$>0.12 or bad_{rate}>0.10, set $G = 0$; else $G = 1$
- $S = 0.35 * S_N + 0.35 * S_Q + 0.30 * S_C$
3) Choose image-base parameter tier P_{base}:
- $S < 0.40$ -> small 15M~35M
- $0.40 <= S < 0.70$ -> medium 35M~80M
- $S >= 0.70$ -> large 80M~160M
4) Apply modality coefficient to get P_{target}:
- image: $k = 1.0$, $P_{target} = P_{base}$

- video: $k = 2.5$, $P_{target} = 2.5 * P_{base}$
- SMPL-H: $k = 0.6$, $P_{target} = 0.6 * P_{base}$

5) Map to your current 32x32 U-Net options:
- small: − − unet − chs 64, 128, 128, 128 − − unet − num − blocks 2 − − unet − no − attn
- medium: − − unet − chs 128, 256, 256, 256 − − unet − num − blocks 2 − − unet − use − attn
- large: − − unet − chs 128, 256, 384, 512 − − unet − num − blocks 3 − − unet − use − attn

Why

Common Issues

choosing model size from sample count only; ignoring video temporal complexity or SMPL-H parameter validity; ignoring alignment noise in text-conditioned training.

Fix

1) If $G = 0$, clean data first and do not move to the large tier.
2) Start from P_{target}, then run a two-tier controlled comparison (for example medium vs large).
3) Freeze sampler/NFE/sample count/random seed during comparison and change model-size tier only (see 5.5-5.6).

A.1 DiT/DyDiT scale scale table (select layers/heads/width) Citation: P5 p17 (Table 7)

How

Selection principle: Let readers know when to use S/B/XL, and be able to reproduce the comparison under the same protocol.
1) When replacing the U-Net of P1 with Transformer (DiT), first lock layers / heads / channel with the following "discrete gear table" to ensure reproducibility and comparison (P5).
2) When dynamizing DyDiT, the static DiT of the same level is still used as the backbone; dynamization only changes the heads/channel groups/tokens activated at each step, and does not change the backbone scale definition (P5).
3) First confirm whether you are in the "covered resolution/task" range of P5:
- The main experiment of P5 is on ImageNet 256x256, and an additional 512x512 is reported (also using DiT-XL as the baseline model) (P5).
- If your target resolution is not 256 or 512: This manual does not claim which gear is "optimal"; you can still use S/B/XL as a discrete sweep, but the conclusion belongs to your new experiment and needs to be evaluated according to the fixed protocol alignment in Chapter 5 before drawing a conclusion.
4) Default decision (strictly align with the "data scarcity/budget/control" context of P4/P5, without introducing extra-paper thresholds):
- ImageNet-256/512 and pursuing the best quality: choose DiT-XL (the strongest static baseline scale reported by P5).
- ImageNet-256/512, but more concerned about fast iteration/passing the closed loop first: choose DiT-S; after running through, upgrade to DiT-B/DiT-XL for comparison (P5).
- Fine-tuning/migration/scarce data: Start with DiT-S first; P4 emphasizes that scarce tasks are more likely to be overfitted/biased, and P5 also uses small models as a common starting point in the context of fine-tuning in the appendix; use the snapshot protocol of 5.6 to confirm overfitting before upgrading.
5) Structure gear table (direct reproduction, from P5 Table 7; purpose: as long as you declare "which gear is used", readers will know your depth/heads/width):

model	params (M)	layers	heads	channel (width)	pre-training source
DiT-S	33	12	6	384	5M iter (Pan et al., 2024)
DiT-B	130	12	12	768	1.6M iter (Pan et al., 2024)
DiT-XL	675	28	16	1152	7M iter (Peebles & Xie, 2023)

6) For a more complete table of "control variables", see the general table A.1a below: fixed model and resolution, scan only λ, and give the FLOPs and FID compared with the same table (P5 Table 8).

Why

The P5 method is to perform dynamic calculations on the existing DiT gears; therefore, if these gears are not used, the conclusions cannot be aligned with the paper.

Common Issues

Fix

Lock the gear first (select S/B/XL), and use resolution and token/patch as separate design variables to be changed in independent experiments. Do not mix them into the "model scale definition".

A.1a Control variable list: Fixed ImageNet-256 and evaluation link, scan target FLOPs ratio by backbone `λ` Citation: P5 p18 (Table 8)

How

1) This is the "single entry summary list" after merging the original A.1a/A.1b/A.1c: fixed resolution = 256x256, fixed evaluation = FID, and scan λ under the three-speed backbone of DiT-S / DiT-B / DiT-XL to uniformly look at the speed-quality tradeoff (P5).
2) When writing a paper/report, you can directly quote this summary table for horizontal comparison: compare the impact of λ within the same backbone, and compare the "scale-efficiency-quality" relationship between different backbones.
Table (P5 Table 8; Note: P5 shows that it uses the optimal batch of each model for batched inference on V100 32G; the actual FLOPs of DyDiT will fluctuate near the target λ):

model	λ	s/image (↓)	acceleration (↑)	FLOPs (G) (↓)	FID (↓)	FID Δ (↓)
DiT-S (static)	1.0	0.65	1.00	6.07	21.46	+0.00
DyDiT-S	0.9	0.63	1.03	5.72	21.06	-0.40
DyDiT-S	0.8	0.56	1.16	4.94	21.95	+0.49
DyDiT-S	0.7	0.51	1.27	4.34	23.01	+1.55
DyDiT-S	0.5	0.42	1.54	3.16	28.75	+7.29
DyDiT-S	0.4	0.38	1.71	2.63	36.21	+14.75
DyDiT-S	0.3	0.32	2.03	1.96	59.28	+37.83
DiT-B (static)	1.0	2.09	1.00	23.02	9.07	+0.00
DyDiT-B	0.9	1.97	1.05	21.28	8.78	-0.29
DyDiT-B	0.8	1.76	1.18	18.53	8.79	-0.28
DyDiT-B	0.7	1.57	1.32	16.28	9.40	+0.33
DyDiT-B	0.5	1.22	1.70	11.90	12.92	+3.85
DyDiT-B	0.4	1.06	1.95	9.71	15.54	+6.47
DyDiT-B	0.3	0.89	2.33	7.51	23.34	+14.27
DiT-XL (static)	1.0	10.22	1.00	118.69	2.27	+0.00
DyDiT-XL	0.7	7.76	1.32	84.33	2.12	-0.15
DyDiT-XL	0.6	6.86	1.49	67.83	2.18	-0.09
DyDiT-XL	0.5	5.91	1.73	57.88	2.07	-0.20
DyDiT-XL	0.3	4.26	2.40	38.85	3.36	+1.09

Why

The core goal of P5 is to "use dynamic calculations to reduce FLOPs without damaging quality as much as possible"; centralized display in a general table makes it easier to make horizontal comparisons under the same protocol.

Common Issues

Fix

First fix the sampling and evaluation links according to the comparison method of P5, and only scan λ; open a new set of experiments for changes in the sampling link (see Chapter 5).

A.2 DyDiT training details table (optimizer/batch/iteration/λ grid) Citation: P5 p17 (Table 6)

How	1) When adding routers to static DiT for DyDiT fine-tuning, use this table to directly give the default starting point of the "fine-tuning training protocol" (optimizer and lr, global batch, warmup, λ grid). Table (P5 Table 6, ImageNet fine-tuning settings): - optimizer: AdamW, learning rate=1e-4 - global batch size: 256 - target FLOPs ratio λ: - DiT-S/DiT-B: {0.9, 0.8, 0.7, 0.5, 0.4, 0.3} - DiT-XL: {0.7, 0.6, 0.5, 0.3} - fine-tuning iterations: - DiT-S: 50,000 - DiT-B: 100,000 - DiT-XL: 150,000 (for λ=0.7), 200,000 (for others) - warmup iterations: DiT-XL uses 30,000 (S/B are 0) - augmentation: random flip - cropping size: 224x224
Why	
Common Issues	
Fix	First run through the λ grid and warmup according to the table, and then discuss more complex stabilization strategies (see 4.4).

A.3 EDM random sampling parameter grid (directly used to search for S_churn, etc.) Citation: P3 p43 (Table 5)

How	1) When doing EDM random sampling, directly use this table to define the search space to avoid incomparability caused by self-created ranges. Table (P3 Table 5, grid search sets): - S_churn: 0,10,20,...,100 - S_tmin: 0,0.005,0.01,0.02,...,1,2,5,10 - S_tmax: 0.2,0.5,1,2,...,10,20,50,80 - S_noise: 1.000,1.001,...,1.010
Why	
Common Issues	
Fix	Fix NFE and solver (Heun/deterministic) and then only search for randomness (see 5.3-5.4).

A.4 EDM training hyperparameter table (training duration/batch/lr/ramp-up/EMA/dropout) Citation: P3 p45 (Table 7)

How	1) Reproduce the improved configuration of P3 or use it as a starting point to directly set the training hyperparameters using this table; no need to read the paper. Table (P3 Table 7, excerpt of key fields; units are based on the caliber of the paper): - CIFAR-10 (Ours): duration=200 Mimg, GPUs=8, batch=512, lr=1e-3, LR ramp-up=10 Mimg, EMA half-life=0.5 Mimg, dropout=13%, dataset x-flips=off, augment prob=12% - ImageNet (Ours): duration=2500 Mimg, GPUs=32, batch=4096, lr=1e-4, LR ramp-up=10 Mimg, EMA half-life=50 Mimg, dropout=10%
Why	
Common Issues	
Fix	First run through the baseline according to the table, and then do minimum ablation (only change one hyperparameter at a time).

A.5 EDM network structure details table (number of residual blocks/attention resolution/heads) Citation: P3 p46 (Table 8)

How	1) When reproducing the P3 network family (DDPM++/NCSN++/ADM), use this table to write " resblocks and attention insertion points" as a definite configuration to avoid implementation inconsistencies. Table (P3 Table 8, excerpt of key fields): - Residual blocks per resolution: DDPM++(VP)=4, NCSN++(VE)=4, ADM(ImageNet)=3 - Attention resolutions: - DDPM++(VP): {16} - NCSN++(VE): {16} - ADM(ImageNet): {32,16,8} - Attention heads: - DDPM++(VP): 1 - NCSN++(VE): 1 - ADM(ImageNet): 6-9-12 (according to different resolution segments)
Why	
Common Issues	
Fix	Write the attention resolution set and resblocks per resolution into config, and print them in the log.

A.6 iDDPM ’ s U-Net scale range and lr scaling rules Citation: P1 p8, p11

How	1) When doing U-Net scaling, use the 4th channel number of P1 as the only gear, and adjust it in conjunction with its lr scaling rules. Table (P1 scaling anchor): - C: 64/96/128/192 - Params: 30M / 68M / 120M / 270M - lr rule: baseline(C=128) lr=1e-4, others use lr = 1e-4 / sqrt(C/128)
Why	
Common Issues	Randomly selecting the number of intermediate channels leads to inability to compare; changing only the model without changing lr leads to convergence and quality distortion.
Fix	Only allow gear jumps, and write the lr rules into the script.

A.7 Two rules for time point subsequence (τ) of DDIM Citation: P2 p6-p8

How	1) Do few-step sampling, use these two rules to generate τ subsequences, and use fixed x_T for consistency diagnosis (see 5.2). Table (P2 τ rules): - Linear: tau_i = floor(c * i) - Quadratic: tau_i = floor(c * i^2) - P2 empirical selection: CIFAR-10 is more quadratic, and the remaining data sets are more linear (as a starting point).
Why	P2 regards the "sampling time point distribution" as a few-step quality critical degree of freedom.
Common Issues	
Fix	First use deterministic DDIM with eta=0 for run-through consistency diagnosis, and then introduce randomness (eta>0).

A.8 Direct cost estimation tips for OCM (why amortization is needed) Citation: P6 p3-p4

How	1) If you want to improve the quality/likelihood in a few steps and consider learning diagonal covariance, first use this cost prompt to determine "whether direct estimation is feasible." Cost Tip (P6): To obtain the ideal diagonal approximation on CIFAR-10, M>=100 Rademacher samples are required; each sample requires one forward + one back pass, the total cost is about 2M network evaluations, and inference is not available.
Why	
Common Issues	Doing Hessian/diagonal estimation directly at inference time, speed is not available.
Fix	Use the OCM head to amortize learning during the training period, and use one forward output diagonal term during the inference period (see 3.7/5.8).

Copyright and usage statement

References (only 6)

- [P1] Alex Nichol, Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672, 2021.

- [P2] Jiaming Song, Chenlin Meng, Stefano Ermon. Denoising Diffusion Implicit Models. arXiv:2010.02502, 2021.

- [P3] Tero Karras, Miika Aittala, Timo Aila, Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. NeurIPS 2022. arXiv:2206.00364.

- [P4] Hang Chen, Qian Xiang, Jiaxin Hu, Meilin Ye, Chao Yu, Hao Cheng, Lei Zhang. Comprehensive exploration of diffusion models in image generation: a survey. Artificial Intelligence Review, 58:99, 2025. DOI: 10.1007/s10462-025-11110-3.

- [P5] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, Yang You. Dynamic Diffusion Transformer. ICLR 2025.

- [P6] Zijing Ou, Mingtian Zhang, Andi Zhang, Tim Z. Xiao, Yingzhen Li, David Barber. Improving Probabilistic Diffusion Models with Optimal Diagonal Covariance Matching. ICLR 2025. arXiv:2406.10808.