Supplementary Material: 2ndMatch: Finetuning Pruned Diffusion Models via Second-Order Jacobian Matching

Caleb Zheng
University of Washington
zheng94@uw.edu

Eli Shlizerman
University of Washington
shlizee@uw.edu

Contents

A. Mathematical Derivation

A.1. 1st-order Jacobian Matching

In this section, we provide a detailed explanation of first-order Jacobian matching (JM) and its equivalence to matching noisy soft targets. The derivation follows the formulation introduced in [15], where JM is used as a regularization strategy to transfer fine-grained input–output sensitivity from a teacher model to a student model in the context of image classification.

We begin with a general function $f : \mathbb{R}^n \to \mathbb{R}^m$. The first-order Taylor expansion of f around a point $x \in \mathbb{R}^n$, within a small perturbation neighborhood $\{x + v : ||v|| < \epsilon\}$, yields

$$f(x+v) = f(x) + \nabla_x f(x)v + O(\epsilon^2). \tag{1}$$

When f represents a neural network, this linear approximation describes how its outputs change under small input perturbations. Now consider the squared error loss between a teacher model $s_{\mathcal{D}}(x)$ and a student model $s(x)$:

$$\ell(s(x), s_{\mathcal{D}}(x)) = \left\|s(x) - s_{\mathcal{D}}(x)\right\|_2^2 \tag{2}$$

$$= \sum_{i=1}^{m} \left(s^i(x) - s^i_{\mathcal{D}}(x)\right)^2. \tag{3}$$

To account for nearby perturbations, let $v = \sigma z$, where $z \in \mathbb{R}^n$ is a unit normal random variable, and $\sigma \in \mathbb{R}$ is a scaling factor. The expected loss under such perturbations becomes

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_v\left[\left\|s(x+v) - s_{\mathcal{D}}(x+v)\right\|_2^2\right]. \tag{4}$$

We apply the first-order Taylor expansion to both the teacher and the student

$$s_{\mathcal{D}}(x+v) = s_{\mathcal{D}}(x) + J_{\mathcal{D}}(x)v + \mathcal{O}(\sigma^2), \tag{5}$$

$$s(x+v) = s(x) + J(x)v + \mathcal{O}(\sigma^2), \tag{6}$$

where $J_{\mathcal{D}}, J \in \mathbb{R}^{m\times n}$ are the teacher's and student's Jacobians of the output with respective to the input, respectively.

The loss becomes

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_v\Big[\sum_{i=1}^{m}\Big(s^i(x) + J^i(x)v - s^i_{\mathcal{D}}(x) - J^i_{\mathcal{D}}(x)v\Big)^2 + \mathcal{O}(\sigma^4)\Big] \tag{7}$$

$$= \sum_{i=1}^{m}\Big(s^i(x) - s^i_{\mathcal{D}}(x)\Big)^2 + \mathbb{E}_v\Big[\sum_{i=1}^{m}\Big((J^i(x) - J^i_{\mathcal{D}}(x))v\Big)^2\Big] + \mathbb{E}_v\Big[\sum_{i=1}^{m}2\Big(s^i(x) - s^i_{\mathcal{D}}(x))(J^i(x) - J^i_{\mathcal{D}}(x)\Big)v\Big] + \mathcal{O}(\sigma^4). \tag{8}$$

The cross term in Eq. 8 vanishes since $\mathbb{E}[v] = 0$ and the high order error $\mathcal{O}(\sigma^4)$ is ignored for small σ. For isotropic Gaussian noise $v = \sigma z$, we have $\mathbb{E}[vv^\top] = \sigma^2 I$, and thus

$$\mathcal{L}_{\text{noisy}} \approx \|s(x) - s_{\mathcal{D}}(x)\|_2^2 + \sigma^2\|J(x) - J_{\mathcal{D}}(x)\|_F^2, \tag{9}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

This expansion reveals that matching the teacher with the student on noisy inputs naturally decomposes into two terms: (i) an *output-level matching loss* $\|s(x) - s_{\mathcal{D}}(x)\|_2^2$, and (ii) a *Jacobian matching loss* $\|J(x) - J_{\mathcal{D}}(x)\|_F^2$,

The Jacobian term penalizes discrepancies in local input–output sensitivity, encouraging the student to preserve the teacher's behavior in a neighborhood around x. In this sense, first-order JM can be interpreted as the leading-order correction induced by training on noisy soft targets.

In diffusion models, the training inputs x_t are *inherently noisy*, as they are generated by a forward noising process from clean data x_0. At each timestep t, the model is trained on corrupted samples of the form

$$x_t = \alpha_t x_0 + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{10}$$

so the standard denoising objective already matches teacher and student predictions over a broad distribution of perturbed inputs across timesteps. From the perspective of the derivation above, this repeated matching over noisy x_t implicitly *encourages* first-order sensitivity alignment, making an explicit first-order JM term potentially redundant while adding non-trivial computational cost (e.g., Jacobian–vector products). This motivates moving beyond first-order alignment to *second-order* Jacobian-based objectives that capture temporal sensitivity and long-range perturbation growth along the denoising trajectory.

To empirically verify whether explicit first-order JM provides non-redundant supervision beyond standard diffusion losses, we ablate first-order JM on both low-resolution (CIFAR-10) and high-resolution (ImageNet) settings. As summarized in Table 1 (and Table. 4 in main manuscript), first-order JM does not improve over the NP+KD baseline and can slightly degrade performance. On CIFAR-10, FID changes from 5.05 (NP+KD) to 5.14 (+1st JM), whereas our second-order matching (2ndM) improves FID to 4.58.

FID ↓	NP	NP+KD	+1st JM	+2ndM (Ours)	Dense
CIFAR-10	5.29	5.05	5.14	**4.58**	4.19
ImageNet	10.23	7.96	8.15	**5.68**	3.60

Table 1. Loss component ablation on CIFAR-10 and ImageNet (LDM-4). Adding 1st-order JM does not improve FID over NP+KD, whereas 2ndM yields a large gain.

A similar trend holds on ImageNet (LDM-4): FID changes from 7.96 (NP+KD) to 8.15 (+1st JM), while 2ndM yields a substantial gain (7.96 → 5.68), narrowing the gap to the dense teacher (FID 3.60). Overall, these results support our claim that first-order alignment is largely redundant in diffusion training, whereas second-order sensitivity matching provides complementary and more effective supervision.

A.2. FTLE

This section provides additional detail on the motivation and derivation of Finite Time Lyapunov Exponents (FTLE). The notation and examples are adapted from [12].

We consider a general time-dependent dynamical system defined by the Ordinary Differential Equation (ODE)

$$\frac{dx(t)}{dt} = f(x(t), t), \tag{11}$$

where $x \in \mathbb{R}^n$ denotes the system state, and $t \in \mathbb{R}$ denotes time. The function $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ governs the evolution of the system and is assumed to be smooth.

Given an initial time t_0 and interval t_1, we define the flow map $\Phi \in \mathbb{R}^n \to \mathbb{R}^n$, which transports a point x_0 to its location at time $t_0 + t_1$, i.e., x_1.

$$\Phi_{t_0}^{t_0+t_1} : x_0 \to x_1 \tag{12}$$

$$= x_0 \to x_0 + \int_{t_0}^{t_0+t_1} f(x(\tau), \tau)d\tau. \tag{13}$$

By standard existence and uniqueness results [7], this map satisfies

$$\Phi_{t_0}^{t_0}(x_0) = x_0, \tag{14}$$

$$\Phi_{t_0}^{t_0+t_1+t_2}(x) = \Phi_{t_0+t_1}^{t_0+t_1+t_2}(\Phi_{t_0}^{t_0+t_1}(x)). \tag{15}$$

While numerical integration of f yields particle trajectories, analyzing time-dependent systems directly can be challenging since studying the system's overall behavior requires integrating many trajectories, which is computationally expensive. FTLE offers a principled way to quantify the rate at which nearby trajectories diverge, revealing coherent structures and deformation patterns in the dynamics. FTLE is more representative of the true transport behavior in a time-dependent system than instantaneous separation

criteria such as the Okubo-Weiss criterion [10, 16] since it reflects the average or accumulated deformation of the flow.

The FTLE, denoted $\Lambda_{t_0}^{t_0+t_1}(x)$, is a scalar value, characterizing how infinitesimal perturbations to a point $x \in \mathbb{R}^n$ grow under the flow map $\Phi_{t_0}^{t_0+t_1}$ over a finite time interval $[t_0, t_0+t_1]$. Specifically, to analyze this behavior, we consider the evolution of two nearby trajectories. Let $x_0 \in \mathbb{R}^n$ be a reference point at time t_0, and let $x_0' = x_0 + v_0$ be a neighboring point, where v_0 is an infinitesimal perturbation. Since the flow is continuously dependent on initial conditions, the trajectory of x' will remain close to that of x, at least for a short time. After t_1, the separation of x and x' becomes

$$v_1 = \Phi_{t_0}^{t_0+t_1}(x_0) - \Phi_{t_0}^{t_0+t_1}(x_0'). \tag{16}$$

Applying the first-order Taylor expansion of the flow map $\Phi_{t_0}^{t_0+t_1}$ around x_0, we obtain

$$v_1 = \frac{d\Phi_{t_0}^{t_0+t_1}(x_0)}{dx_0} v_0 + O(||v_0||^2). \tag{17}$$

Since v_0 is infinitesimal, the higher-order term $O(||v_0||^2)$ can be neglected. Therefore, the norm of the perturbation after t_1 becomes

$$||v_1|| \approx ||\frac{d\Phi_{t_0}^{t_0+t_1}(x_0)}{dx_0} v_0|| \tag{18}$$

$$= \sqrt{v_0^\top \mathcal{C}\, v_0}, \tag{19}$$

where $\mathcal{C} \in \mathbb{R}^{n\times n}$ is the finite-time Cauchy-Green deformation tensor, defined as

$$\mathcal{C} = J^\top J, \tag{20}$$

$$J = \frac{d\Phi_{t_0}^{t_0+t_1}(x_0)}{dx_0}. \tag{21}$$

Here, $(\cdot)^\top$ denotes the matrix transpose operation. The matrix $\mathcal{C}$ captures the accumulated stretching and compression of infinitesimal vectors under the flow over the time interval t_1. To quantify the maximum stretching, we align v_0 with the eigenvector corresponding to the largest eigenvalue $\lambda_{\max}(\mathcal{C})$,

$$\max_{v_0} ||v_1|| = \sqrt{\lambda_{max}(\mathcal{C})}||v_0||. \tag{22}$$

Consequently, the FTLE is defined as

$$\Lambda_{t_0}^{t_0+t_1}(x) = \frac{1}{t_1} \ln \sqrt{\lambda_{max}(\mathcal{C})}, \tag{23}$$

The FTLE quantifies the local exponential rate of trajectory separation and has been commonly used to identify coherent structures and transport barriers in dynamical systems [6].

A.3. FTLE for Discrete-Time Diffusion Denoising Dynamics

Diffusion sampling is naturally a *discrete-time* dynamical system: each denoising step maps a latent (or image) state at step k to the next state $k+1$. This section reformulates FTLE in that setting using the same flow-map and Cauchy–Green notation as in Sec. A.2.

Discrete denoising dynamics and flow map. Let a diffusion sampler define an iterative update

$$x_{k+1} = \Phi_k(x_k; \theta), \qquad k = 0, 1, \ldots, K-1, \tag{24}$$

where $x_k \in \mathbb{R}^n$ is the state at step k (e.g., latent/image), θ are model parameters, and Φ_k denotes one denoising update (e.g., DDPM/DDIM/Euler step) at the corresponding noise level. For an initial step k_0 and horizon m, define the m-step flow map

$$\Phi_{k_0:k_0+m}(x_{k_0}; \theta) \;\coloneqq\; \Phi_{k_0+m-1}(\cdot; \theta) \circ \cdots \circ \Phi_{k_0}(\cdot; \theta)\,(x_{k_0}), \tag{25}$$

which transports x_{k_0} to x_{k_0+m}. It satisfies the composition property

$$\Phi_{k_0:k_0}(x; \theta) = x, \tag{26}$$

$$\Phi_{k_0:k_0+m_1+m_2}(x; \theta) = \Phi_{k_0+m_1:k_0+m_1+m_2}(\Phi_{k_0:k_0+m_1}(x; \theta); \theta)\,. \tag{27}$$

Consider two nearby initial states at step k_0: $x'_{k_0} = x_{k_0} + v_{k_0}$ with an infinitesimal perturbation v_{k_0}. After m denoising steps,

$$v_{k_0+m} \;=\; \Phi_{k_0:k_0+m}(x_{k_0} + v_{k_0}; \theta) - \Phi_{k_0:k_0+m}(x_{k_0}; \theta). \tag{28}$$

Linearizing the m-step flow map around x_{k_0} yields

$$v_{k_0+m} \approx J_{k_0:k_0+m}(x_{k_0}; \theta)\, v_{k_0}, \tag{29}$$

where the m-step Jacobian is

$$\begin{aligned} J_{k_0:k_0+m}(x_{k_0}; \theta) \;&\coloneqq\; \frac{\partial \Phi_{k_0:k_0+m}(x_{k_0}; \theta)}{\partial x_{k_0}} \\ &= \prod_{i=0}^{m-1} J_{k_0+i}(x_{k_0+i}; \theta), \end{aligned} \tag{30}$$

$$J_k(x_k; \theta) \coloneqq \frac{\partial \Phi_k(x_k; \theta)}{\partial x_k}. \tag{31}$$

The product in (30) follows from the chain rule, since the overall sensitivity from x_{k_0} to x_{k_0+m} accumulates multiplicatively across intermediate denoising steps.

The finite-time Cauchy–Green deformation for diffusion flow is defined as

$$\mathcal{C}_{k_0}^{k_0+m}(x_{k_0};\theta) \;\coloneqq\; J_{k_0:k_0+m}(x_{k_0};\theta)^\top\, J_{k_0:k_0+m}(x_{k_0};\theta). \tag{32}$$

Then the perturbation norm after m steps satisfies

$$\|v_{k_0+m}\| \approx \left\|J_{k_0:k_0+m} v_{k_0}\right\| = \sqrt{\langle v_{k_0},\, \mathcal{C}_{k_0}^{k_0+m}\, v_{k_0}\rangle}. \tag{33}$$

The maximal stretching over all unit perturbations is

$$\max_{\|v_{k_0}\|=1} \|v_{k_0+m}\| \approx \sqrt{\lambda_{\max}\Big(\mathcal{C}_{k_0}^{k_0+m}(x_{k_0};\theta)\Big)}. \tag{34}$$

The discrete FTLE at state x_{k_0} over m denoising steps is defined as the average exponential separation rate per step:

$$\Lambda_{k_0}^{k_0+m}(x_{k_0};\theta) \;\coloneqq\; \frac{1}{m}\,\ln\sqrt{\lambda_{\max}\Big(\mathcal{C}_{k_0}^{k_0+m}(x_{k_0};\theta)\Big)}. \tag{35}$$

Relationship between single-step and multi-step Jacobian mismatch. Directly matching the full multi-step tensor $\mathcal{C}_{k_0}^{k_0+m}$ is expensive, as it requires Jacobian products across m steps. In practice, we use a single-step matching by setting m=1, for which

$$\mathcal{C}_k^{k+1}(x_k;\theta) = J_k(x_k;\theta)^\top J_k(x_k;\theta), \tag{36}$$

This motivates our focus on matching second-order sensitivity statistics derived from $J_k^\top J_k$ at individual denoising steps, which empirically captures much of the relevant local stretching behavior while remaining tractable.

To further justify the single-step matching, consider a teacher ($s_{\mathcal{D}}$) and a student (s) dynamics with Jacobians $\{J_{k+i}^{\mathcal{D}}\}$ and $\{J_{k+i}\}$, respectively. Using the product-difference identity, the m-step Jacobian mismatch can be decomposed as

$$J_{k:k+m} - J_{k:k+m}^{\mathcal{D}} = \sum_{j=0}^{m-1}\Big(\prod_{i=j+1}^{m-1} J_{k+i}\Big)\,(J_{k+j} - J_{k+j}^{\mathcal{D}})\Big(\prod_{i=0}^{j-1} J_{k+i}^{\mathcal{D}}\Big). \tag{37}$$

Taking an operator norm and applying $\|AB\| \le \|A\|\,\|B\|$ yields

$$\begin{aligned}\|J_{k:k+m} - J_{k:k+m}^{\mathcal{D}}\| \;\le\; \sum_{j=0}^{m-1}\Big(&\Big\|\prod_{i=j+1}^{m-1} J_{k+i}\Big\|\cdot \\ &\|J_{k+j} - J_{k+j}^{\mathcal{D}}\|\cdot\Big\|\prod_{i=0}^{j-1} J_{k+i}^{\mathcal{D}}\Big\|\Big).\end{aligned} \tag{38}$$

The multi-step Jacobian mismatch is upper bounded by a weighted sum of per-step mismatches, with weights determined by intermediate Jacobian products. Consequently, some steps may be more influential than others for the accumulated multi-step sensitivity mismatch. A step-wise matching objective, by contrast, promotes local alignment across all steps more uniformly, rather than allowing the overall mismatch to be dominated by only a few influential steps. In this sense, step-wise matching imposes a stronger constraint than matching only the composed multi-step behavior.

# Steps	GPU Memory ↓	Training Time ↓	FID ↓
1	34 GB	18 hr	4.58
2	80 GB (×2.35)	40 hr (×2.2)	4.62 (+0.04)
3	120 GB (×3.53)	60 hr (×3.3)	4.55 (−0.03)

Table 2. Cost–accuracy trade-off of multi-step sensitivity matching on CIFAR-10. Multi-step variants provide comparable FID to the single-step objective but require substantially higher memory and training time.

Empirical cost–accuracy trade-off (CIFAR-10). We additionally tested explicit multi-step variants. As shown in Table 2, multi-step matching achieves *comparable* FID to the single-step objective but with substantially higher compute and memory costs that scale roughly linearly with the number of matched steps. Given the strong cost scaling and limited gains, we adopt the step-wise matching in the main method.

B. FTLE Computation for Diffusion Models

For the FTLE results reported in Table 2 of the main manuscript, we evaluate the temporal sensitivity of each model after finetuning has been completed. All models—NP, NP+KD, +1st JM, +2ndM, and the dense teacher—are evaluated using the same fixed batch of 128 noise samples at CIFAR-10 resolution ($3 \times 32 \times 32$). These shared inputs ensure that differences in FTLE arise solely from differences in the models rather than the initial conditions.

We use a DDIM sampler with 100 denoising steps for generating trajectories. However, computing the full Jacobian at each of these steps is infeasible even for CIFAR-10–sized models: backpropagating through the UNet across many consecutive timesteps builds up large computation graphs and quickly exhausts GPU memory. Due to these constraints, we compute Jacobians only at the first 10 DDIM timesteps—the maximum number our hardware can support without running out of memory. Using early-stage timesteps is also meaningful, as these steps exhibit the highest sensitivity and contribute most strongly to perturbation growth.

While 2ndM relies on random-projection–based second-order Jacobian supervision during training—a necessity because obtaining full Jacobians at every step of training

Table 3. **Key training and architectural hyperparameters for DDPM, LDM–4 (Cin256-v2), and Stable Diffusion 1.4.**

Dataset	Model	Res.	Batch	LR	Linear β	T	Channels
CIFAR–10	DDPM	32^2	128	2×10^{-4}	$0.0001 \rightarrow 0.02$	1000	$[128, 256, 256, 256]$
LSUN	DDPM	256^2	64	2×10^{-5}	$0.0001 \rightarrow 0.02$	1000	$[128, 128, 256, 256, 512, 512]$
ImageNet	LDM–4	256^2 (latent 64^2)	128	2.56×10^{-4}	$0.0015 \rightarrow 0.0195$	1000	$[192, 384, 576, 960]$
MS–COCO	SD v1.4	512^2 (latent 64^2)	256	5×10^{-5}	$0.00085 \rightarrow 0.012$	1000	$[320, 640, 1280, 1280]$

Table 4. **Key training and architectural hyperparameters for U-ViT-Small** on CIFAR-10 and CelebA.

Dataset	Model	Res.	Batch	LR	Warmup	T	Embed dim	Patch	Depth	Heads	mlp ratio
CIFAR–10	U-ViT-Small	32^2	128	2×10^{-4}	2500	1000	512	2	12	8	4
CelebA	U-ViT-Small	64^2	128	2×10^{-4}	5000	1000	512	4	12	8	4

would be prohibitively expensive—our FTLE analysis uses the entire Jacobian matrix. This distinction is important: while random projections provide a scalable approximation for learning, FTLE requires the exact deformation tensor $J_t^T J_t$, which can only be computed from the full Jacobian. Thus, the FTLE numbers in Table 2 reflect the true sensitivity of each model and are not affected by the random-projection approximation used during training.

C. Experimental Setup Details

C.1. Package version

We adopt Diff-Pruning [5] as our primary pruning method. Since Diff-Pruning relies on *torch-pruning* library [4], we use version 1.5.2 for all Transformer-based experiments. This version includes full support for structural pruning in Transformer architectures. For U-Net–based diffusion models, however, we use the earlier 1.2.5 release. The newer versions modify the pruning behavior for U-Net blocks, which leads to different parameter and MAC counts compared to the original Diff-Pruning implementation. To ensure strict comparability with prior work and to maintain consistency across baselines, we retain the original pruning setup for all U-Net experiments.

C.2. Hyperparameters

We provide detailed architectural and training hyperparameters for both U-Net–based and Transformer-based (U-ViT) diffusion models used in our experiments.

U-Net Models. For U-Net architectures, we follow widely adopted configurations from prior diffusion model literature [5, 8, 9, 11]. These settings are applied across CIFAR-10, LSUN, ImageNet, and MS-COCO. Table 3 reports the key hyperparameters for DDPM, LDM–4 (CIN256-v2), and Stable Diffusion v1.4, including input resolution, batch size, learning rate, linear β schedule, total number of diffusion steps, and channel widths.

Transformer-Based Models. For Transformer-based diffusion architectures, we adopt the U-ViT design [3] and evaluate it on CIFAR-10 and CelebA. Table 4 summarizes both training hyperparameters (optimizer, learning rate schedule, warmup, batch size, number of diffusion steps) and architectural details (patch size, embedding dimension, network depth, number of attention heads, and MLP ratio). Following common practice, we train both datasets for 100k steps rather than the full 500k-step schedule used in the original U-ViT paper. This shorter schedule is sufficient because pruned models are initialized from a pretrained dense checkpoint and therefore converge substantially faster than models trained from scratch [13], while still enabling fair comparison across finetuning methods.

D. Computational Cost Analysis

We report the training cost of our proposed 2ndM finetuning framework compared to the Diff-Pruning baseline across both U-Net–based and Transformer-based diffusion models. Tables 5 and 6 summarize batch sizes, GPU memory usage, per-iteration latency, training steps, and total wall-clock training time under matched pruning settings.

For U-Net models (Table 5), we observe that 2ndM introduces a 4–5$\times$ increase in per-iteration training time due to the Jacobian-vector product (JVP) computations required by second-order matching. This overhead translates directly into longer training duration: on CIFAR-10 (pruning ratio 0.44), training time increases from 4.4 to 18.1 hours; on LSUN-Church (pruning ratio 0.30, trained on 8 A100 GPUs), it increases from 72.2 to 391.7 hours. A similar pattern appears on ImageNet and MS-COCO.

In addition to the higher compute cost, 2ndM incurs a substantial memory overhead. Memory usage increases from 8G to 34G on CIFAR-10 and from 8×19G to 8×72G on LSUN-Church, largely due to storing intermediate activations for JVPs and maintaining parallel teacher–student computation graphs. This overhead scales primarily with model size rather than input resolution. For MS-COCO, al-

Table 5. Training cost comparison for UNet-based diffusion models on CIFAR-10, LSUN, ImageNet and MSCOCO.

Dataset	Method	Batch	Memory	Time per Iter	Training Steps	Training Time
CIFAR10	Baseline	128	8 G	0.16 s	100k	4.4 hr
	KD	128	16 G	0.30 s	100k	8.3 hr
	2ndM	128	34 G	0.65 s	100k	18.1 hr
LSUN	Baseline	64	8 × 19 G	0.52 s	500k	72.2 hr
	KD	64	8 × 37 G	1.01 s	500k	140.3 hr
	2ndM	64	8 × 72 G	2.82 s	500k	391.7 hr
ImageNet	Baseline (DP)	128	8 × 13 G	0.34 s	200k	19.3 hr
	KD	128	8 × 26 G	0.53 s	200k	29.4 hr
	2ndM	128	8 × 45 G	0.99 s	200k	56.1 hr
MSCOCO	Baseline (Base)	256	8 × 43G	2.17s	50k	30.2 hr
	Base + KD	256	8 × 65G	3.40s	50k	47.2 hr
	Base + 2ndM	256	8 × 77G	7.06s	50k	98.1 hr
	Baseline (Small)	256	8 × 37G	1.29s	50k	17.8 hr
	Small + KD	256	8 × 55G	2.08s	50k	28.9 hr
	Small + 2ndM	256	8 × 72G	6.76s	50k	94.0 hr
	Baseline (Tiny)	256	8 × 37G	1.29s	50k	18.0 hr
	Tiny + KD	256	8 × 55G	2.08s	50k	28.9 hr
	Tiny + 2ndM	256	8 × 72G	6.76s	50k	94.0 hr

Table 6. Training cost comparison for Transformer-based diffusion models on CIFAR-10 and CelebA

Dataset	Method	Batch	Memory	Time per Step	Training Steps	Training Time
CIFAR10	Baseline	128	11 G	0.13 s	100k	3.6 hr
	2ndM	128	47 G	0.67 s	100k	18.6 hr
CelebA	Baseline	128	11 G	0.43 s	100k	11.9 hr
	2ndM	128	47 G	0.67 s	100k	18.6 hr

though the input resolution is 512^2, the diffusion model operates in a 64^2 latent space; the dominant factor is the significantly larger U-Net backbone, which makes Jacobian computation particularly memory intensive. Consequently, the per-GPU batch size must be reduced to 8, and we employ `accumulate_grad = 4` to match the baseline's effective batch size of 32 per GPU. Gradient accumulation preserves training stability but increases wall-clock time proportionally, explaining the larger training-time gap between 2ndM and the baseline on MS-COCO.

Transformer-based models (Table 6) show a slightly different trend. The U-ViT architecture used for both CIFAR-10 and CelebA is identical in size, so increasing the image resolution slows down both baseline and 2ndM proportionally. As a result, the relative time gap narrows compared to U-Net models: training on CIFAR-10 increases from 3.6 to 18.6 hours, whereas on CelebA the increase is from 11.9 to 18.6 hours. The higher resolution of CelebA increases the baseline cost, reducing the relative overhead introduced by second-order JM.

Overall, the increased training time and memory footprint stem from computing the second-order JM loss. While **2ndM** provides consistent improvements in generative quality, its resource demands may limit applicability in constrained settings. Importantly, this additional cost arises *only during training*: **2ndM** does not modify the model architecture or sampling procedure and therefore introduces *no overhead at inference time*.

E. Effect of KD Strength

We examine whether performance gains are solely attributable to stronger knowledge distillation (KD) supervision by sweeping the KD weight λ_{KD} and comparing KD-only training against KD augmented with 2ndM. Table 7 shows that for every tested λ_{KD}, adding 2ndM consistently improves FID. For instance, at λ_{KD}=1.0, FID improves from 5.03 (KD only) to 4.58 (KD+2ndM). These results indicate that the improvement is not merely due to increasing

λ_{KD}	0.01	0.1	1.0	10.0
FID (KD only) ↓	5.06	5.15	5.03	5.24
FID (KD + 2ndM) ↓	**4.61**	**4.69**	**4.58**	**4.67**

Table 7. Sweep over KD weight λ_{KD} (with λ_{NP}=1). Adding 2ndM consistently improves FID across all KD strengths.

the KD loss weight; rather, 2ndM provides complementary supervision beyond KD.

F. Jacobian Loss Weights vs. FID

2ndM introduces a hybrid finetuning objective composed of three terms:

$$\mathcal{L}_{total} = \lambda_{\text{NP}}\mathcal{L}_{\text{NP}} + \lambda_{\text{KD}}\mathcal{L}_{\text{KD}} + \lambda_{\text{2nd-Jac}}\mathcal{L}_{\text{2nd-Jac}} \quad (39)$$

The relative weight of each component is a tunable hyperparameter. Because an exhaustive hyperparameter search is computationally expensive for large models and high-resolution datasets, we conduct systematic weight sweeps only on CIFAR-10 (U-Net and U-ViT) and CelebA (U-ViT). For all remaining experiments—including LSUN, ImageNet, and MS-COCO—we fix $\lambda_{kd} = 1.0$ and $\lambda_{\text{2nd-Jac}} = 10^{-3}$ which we find to be robust across architectures and datasets.

We analyze how the Jacobian loss weight affects generative performance under different pruning ratios. Experiments are conducted on both U-Net and Transformer-based diffusion models across CIFAR-10 and CelebA.

F.1. U-Net

As shown in Table 8, the Jacobian loss weight has a noticeable impact on FID. Since the Jacobian term is intended to act as a relatively small regularizer on top of the stronger output-level KD supervision, we focus on comparatively small weights for high pruning ratios. Empirically, moderate pruning (e.g., PR $= 0.44$) can benefit from a relatively larger Jacobian weight, yielding substantial gains in FID, whereas more aggressively pruned models favor smaller weights. In these high-pruning regimes, a weaker second-order signal is sufficient to guide the student toward the teacher's sensitivity while avoiding over-constraining a model with limited capacity. Overall, the U-Net results indicate that the Jacobian term should remain significantly smaller than the KD term, and that the preferred weight gradually decreases as the pruning ratio becomes more aggressive.

F.2. U-ViT

We observe similar behavior for the U-ViT backbone on CIFAR-10 and CelebA (Tables 9 and 10). Across both datasets, small to moderate Jacobian weights consistently improve FID over the Diff-Pruning baseline, with the most pruned models again favoring the smallest weights. This pattern aligns with the U-Net experiments: when pruning is mild, a stronger Jacobian regularizer can be beneficial, but under heavy pruning the optimal setting shifts toward smaller values that complement, rather than dominate, the KD objective. Taken together, these studies support the use of $\lambda_{\text{2nd-Jac}} = 10^{-3}$ as a practical default in our large-scale experiments, reflecting the role of the Jacobian term as a light but effective second-order correction on top of the primary KD and noise-prediction losses.

Table 8. Jacobian matching loss weight vs. FID on CIFAR-10 for the UNet-based diffusion model. A weight of 0 corresponds to the Diff-Pruning baseline (no Jacobian supervision).

CIFAR-10 32 × 32						
PR	**FID** (↓) for different weights					
Weight	0	1e-4	1e-3	1e-2	1e-1	1
0	4.19	–	–	–	–	–
0.44	5.29	–	–	4.66	**4.58**	4.93
0.56	6.49	–	5.85	**5.54**	5.66	–
0.70	8.85	7.35	**7.09**	7.16	–	–
0.82	13.42	11.04	**10.91**	11.08	–	–

Table 9. Jacobian matching loss weight vs. FID on CIFAR-10 for the Transformer-based diffusion model. A weight of 0 corresponds to the Diff-Pruning baseline (no Jacobian supervision).

CIFAR-10 32 × 32						
PR	**FID** (↓) for different weights					
Weight	0	5e-4	1e-3	1e-2	5e-2	1e-1
0	3.11	–	–	–	–	–
0.5	3.53	–	3.56	**3.51**	–	4.47
0.7	4.63	–	4.13	**4.05**	5.01	6.04
0.8	6.68	5.07	**4.98**	5.22	–	–

Table 10. Jacobian matching loss weight vs. FID on CelebA for the Transformer-based diffusion model. A weight of 0 corresponds to the Diff-Pruning baseline (no Jacobian supervision).

CelebA (64 × 64)						
PR	**FID** (↓) for different weights					
Weight	0	1e-3	5e-3	1e-2	1e-1	5e-1
dense	2.87	–	–	–	–	–
0.5	3.25	–	–	3.23	**3.21**	–
0.7	3.57	3.32	3.33	**3.30**	3.36	3.46
0.8	4.61	4.10	**4.01**	4.11	–	–

Table 11. **FID results under different OTD hyperparameter settings.** Each column corresponds to a combination of integration step size (Δt), number of OTD update steps (*step*), and probability of applying OTD (*Prob*). The rightmost column reports the FID obtained without OTD, shown for reference.

Δt	0.01						0.1						-
step	1			2			1			2			-
Prob	0.2	0.5	1.0	0.2	0.5	1.0	0.2	0.5	1.0	0.2	0.5	1.0	-
FID	4.66	4.62	4.71	4.63	4.63	4.66	4.64	4.62	4.67	4.63	4.62	4.67	**4.58**

G. Further Potential Improvement

Optimally Time-Dependent (OTD) Random Projection Random projections are widely used to approximate Jacobian behavior in neural network training due to their computational efficiency [14]. However, standard random vectors treat all directions uniformly, without regard to their dynamical relevance. In diffusion models, certain directions—particularly those aligned with unstable modes of the Jacobian—can dominate local dynamics and introduce significant variance. These directions are often associated with large FTLE values, indicating strong local instability and high sensitivity to input perturbations.

Because diffusion models evolve over many iterative steps, the dominant unstable directions can vary over time. To capture this time-varying structure, we explore *Optimally Time-Dependent* (OTD) modes, originally developed for reduced-order modeling in dynamical systems [1, 2]. OTD evolves a low-dimensional basis that tracks transiently unstable directions along a trajectory. Given the Jacobian J, the OTD basis $V = [v_1, v_2, \ldots, v_r] \in \mathbb{R}^{n\times r}$ evolves according to

$$\frac{\partial V}{\partial t} = JV - VV^T JV. \tag{40}$$

In our setting, we adopt the simplest case with a single dominant mode ($r = 1$) to control computational cost. We initialize the basis vector $v \in \mathbb{R}^n$ by sampling from a standard normal distribution and normalizing to unit norm:

$$v_1 \sim N(0, I), \hat{v_1} \leftarrow \frac{v_1}{||v_1||} \tag{41}$$

We then evolve $\hat{v_1}$ over time using the Forward Euler method to numerically integrate the OTD via Eq. 40.

$$\hat{v_1} = \hat{v_1} + \Delta t(J\hat{v_1} - \hat{v_1}\hat{v_1}^\top J\hat{v_1}), \tag{42}$$

where J is the Jacobian at the current timestep and Δt is the integration step size. This update yields a time-adaptive direction $\hat{v}$ that tracks the most unstable local direction at each step.

We then replace the standard random projection used in our second-order JM with this OTD-enhanced direction,

$$\mathcal{L}_{\text{2nd-Jac}} = \|\hat{v}^\top J^\top J\hat{v} - \hat{v}^\top J_{\mathcal{D}}^\top J_{\mathcal{D}}\hat{v}\|_2^2, \tag{43}$$

where J and $J_{\mathcal{D}}$ denote the Jacobians of the pruned and dense (teacher) models, respectively. By aligning projections with the most dynamically relevant direction, this loss provides supervision that is both time-adaptive and informed by the local stability properties of the denoising dynamics. Intuitively, the shared OTD direction encourages the pruned model to match not only the teacher's outputs but also its most sensitive response direction, which could in principle lead to faster convergence and improved generative quality.

Despite this theoretical motivation, our empirical results in Table 11 show that OTD-based projections do not improve FID under our current training setup. Across a range of hyperparameter choices—including different integration step sizes (Δt), the number of OTD updates per training iteration (*step*), and the probability of applying OTD (*Prob*)—the resulting FID scores remain comparable to, or slightly worse than, our standard random-projection baseline. This suggests that while OTD effectively identifies transiently unstable modes of the Jacobian, this additional focus on dynamically dominant modes does not directly translate into perceptual gains in our experiments.

Nonetheless, we view this direction as promising. Replacing uniform random projections with more principled, dynamically informed projection schemes may become beneficial when combined with more accurate integration methods, adaptive schedules over training, or multiple tracked modes ($r > 1$). We leave a more systematic exploration of such OTD-based and other structured projection strategies for future work.

H. Limitation

While our finetuning framework provides an effective way to inject Jacobian-based supervision into pruned diffusion models, several limitations remain and highlight opportunities for future improvement.

Computational Overhead from Jacobian Calculations. Our method introduces additional training cost due to the need to compute Jacobian-vector products, see Table. 5 and 6. While we employ random projections to reduce

the dimensionality of Jacobian matching, the added backward passes still increase training time and GPU memory usage—especially in high-resolution settings or when using large backbone models. This overhead may limit the method's accessibility to groups with substantial computational resources and constrain its scalability to even larger diffusion architectures.

Sensitivity to the Jacobian Loss Weight. Performance depends on choosing an appropriate coefficient for Jacobian loss, especially across different pruning ratios (Tables 8, 9, 10). While moderate pruning often benefits from stronger Jacobian supervision, higher pruning ratios tend to require smaller weights to avoid overregularization. Identifying an optimal weight may therefore require multiple training runs, adding additional tuning cost. In practice, we find that a default Jacobian weight of 10^{-3} works reliably across most datasets and architectures that we works with, but further study is needed to develop more adaptive or self-tuning strategies.

Dependence on Dense Teacher Quality. As with distillation-based finetuning, our approach assumes access to a strong and stable dense teacher. If the teacher is weak or exhibits training instabilities, the student is inherently bounded by the teacher and may inherit suboptimal behaviors or instability patterns through both output-level distillation and Jacobian-based supervision. While our experiments use well performing dense teachers, a systematic study of distillation under weak or unstable teachers (and how to perform Jacobian supervision in such settings) is beyond the scope of this work.

Table 12. Relative FID (rFID) for selected ImageNet classes, sorted by class index.

Class ID	Class Name	DP rFID ↓	Jac rFID ↓
22	bald eagle	12.30	7.14
207	golden retriever	14.15	5.70
292	tiger	20.59	3.01
388	giant panda	18.27	7.41
393	anemone fish	20.47	4.71
654	minibus	11.80	7.67
718	pier	16.75	4.23
976	promontory	18.65	8.56
979	valley	18.91	9.30

I. Additional Quantitative Results on ImageNet

To better understand class-level behavior on ImageNet, Table 12 reports the relative FID for a selection of representative classes, sorted by class index. In all cases, our Jacobian-based finetuning yields substantially lower relative FID compared to Diff-Pruning, indicating closer alignment with the dense teacher model. The improvements span diverse semantic categories—including animals, vehicles, and natural scenes—suggesting that the benefit of Jacobian supervision generalizes across heterogeneous visual concepts.

J. Visualization of Generated Images

We provide additional qualitative comparisons between the baseline and **2ndM** across all datasets evaluated in this work. Representative samples for LSUN-Church and LSUN-Bedroom are shown in Figure 1, while ImageNet, MS-COCO, CIFAR-10, and CelebA results are shown in Figures 2, 3, 4, and 5, respectively.

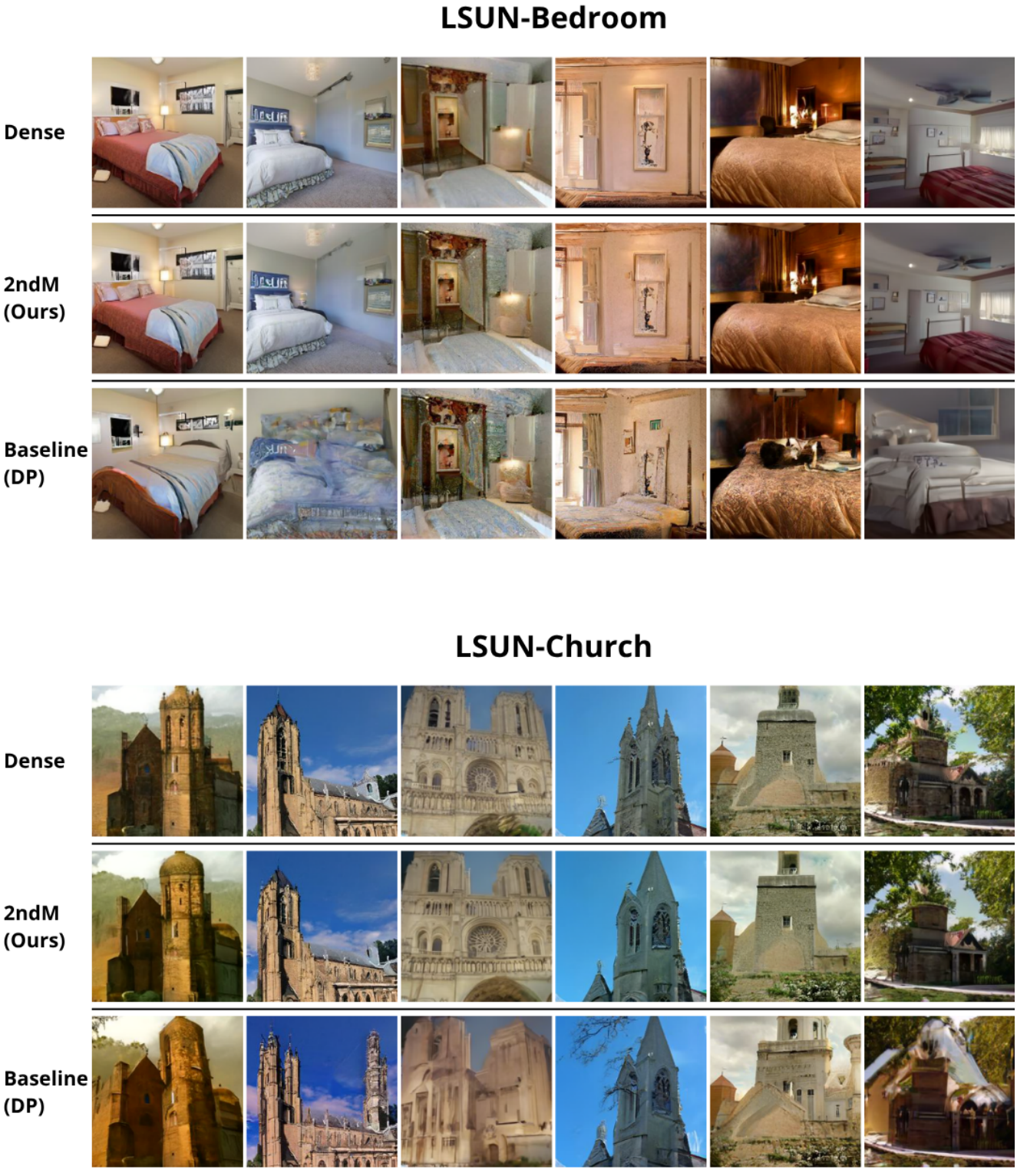

Figure 1. Visual comparison of LSUN-Bedroom and LSUN-Church samples from dense models (top), 2ndM (middle) and baseline Diff-Pruning (bottom).

Figure 2. Visual comparison of ImageNet samples from dense models (top), 2ndM (middle) and baseline Diff-Pruning (bottom).

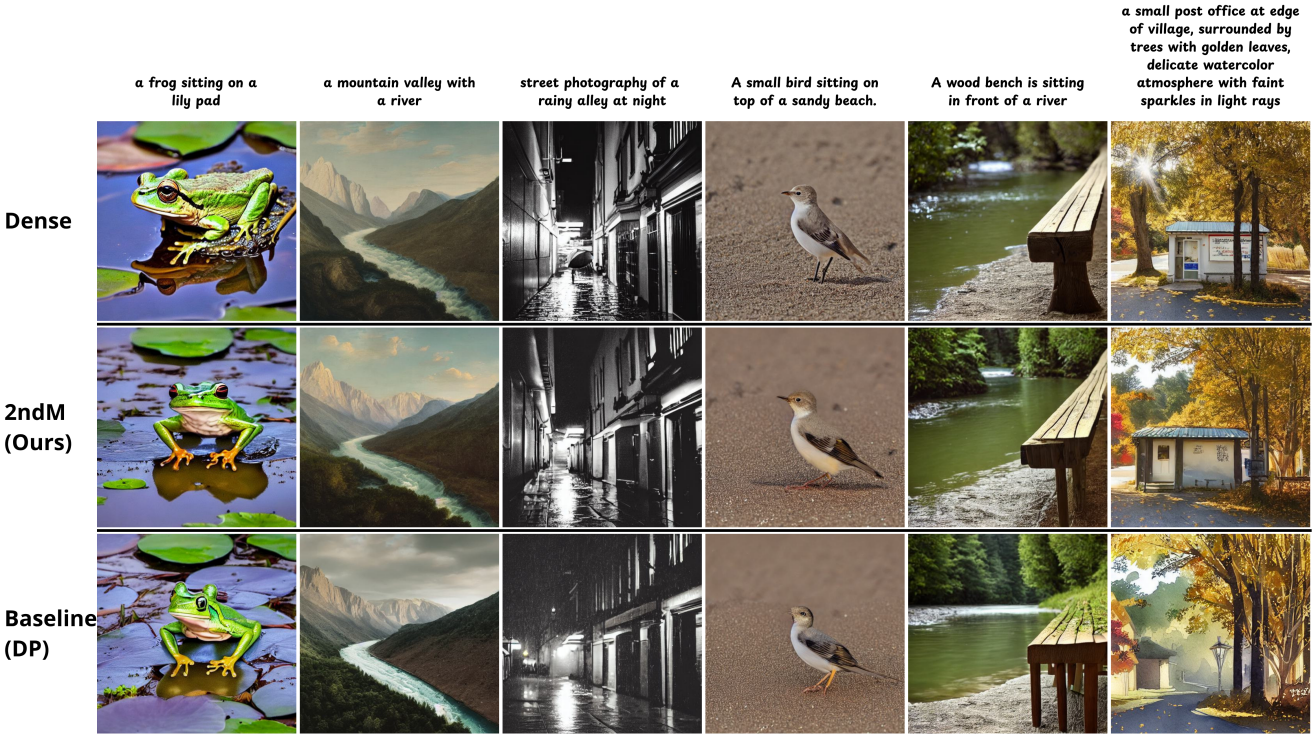

Figure 3. Visual comparison of MSCOCO samples from dense models (top), 2ndM (middle) and baseline Diff-Pruning (bottom).

Figure 4. Visual comparison of CIFAR-10 samples from dense models (top), 2ndM (middle) and baseline Diff-Pruning (bottom).

References

[1] H Babaee and TP Sapsis. A minimization principle for the description of modes associated with finite-time instabilities. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2186):20150779, 2016. 8

[2] Hessam Babaee, Mohamad Farazmand, George Haller, and Themistoklis P Sapsis. Reduced-order description of transient instabilities and computation of finite-time lyapunov exponents. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(6), 2017. 8

[3] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023. 5

[4] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16091–16101, 2023. 5

[5] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *arXiv preprint arXiv:2305.10924*, 2023. 5

[6] George Haller and Guocheng Yuan. Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena*, 147(3-4):352–370, 2000. 3

[7] Philip Hartman. *Ordinary differential equations*. SIAM, 2002. 2

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 5

[9] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. On architectural compression of text-to-

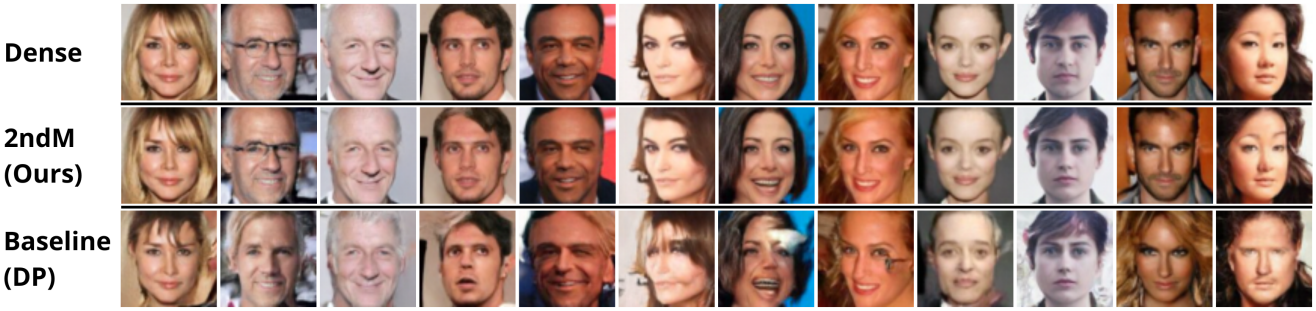

Figure 5. Visual comparison of CelebA samples from dense models (top), 2ndM (middle) and baseline Diff-Pruning (bottom).

image diffusion models. *arXiv preprint arXiv:2305.15798*, 2023. 5

[10] Akira Okubo. Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences. In *Deep sea research and oceanographic abstracts*, pages 445–454. Elsevier, 1970. 3

[11] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 5

[12] Shawn C Shadden. Lagrangian coherent structures. *Transport and mixing in laminar flows: from microfluidics to oceanic currents*, pages 59–89, 2011. 2

[13] Reza Shirkavand, Peiran Yu, Shangqian Gao, Gowthami Somepalli, Tom Goldstein, and Heng Huang. Efficient fine-tuning and concept suppression for pruned diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18619–18629, 2025. 5

[14] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020. 8

[15] Suraj Srinivas and François Fleuret. Knowledge transfer with jacobian matching. In *International Conference on Machine Learning*, pages 4723–4731. PMLR, 2018. 1

[16] John Weiss. The dynamics of enstrophy transfer in two-dimensional hydrodynamics. *Physica D: Nonlinear Phenomena*, 48(2-3):273–294, 1991. 3