ManiFusion: Unlocking the High-Throughput Generation via Superposition in Manifold Space

Minkyu Kim[1] **Baekseung Kim**[1] **Junhoo Lee**[1] **Jangho Kim**[2,*] **Nojun Kwak**[1,*]
[1]Seoul National University [2]Kookmin University
{minkyu.kim,baekseung.kim,mrjunoo,nojunk}@snu.ac.kr
jangho.kim@kookmin.ac.kr

Abstract

Diffusion models achieve high-quality image generation but remain computationally expensive due to iterative denoising across many timesteps. Existing acceleration methods mainly reduce per-sample latency, but gains along this axis are becoming saturated. We instead explore an orthogonal throughput-oriented direction by processing multiple samples within a single denoising evaluation. To this end, we propose ManiFusion, a backbone-agnostic framework for multi-sample generation through fused-space computation. ManiFusion jointly processes multiple samples by fusing them before the denoising backbone and separating them afterward through unfusion. We provide a geometric perspective on how multiple samples can share a fused representation while retaining the information needed to guide their individual generation trajectories. To better align fusion allocation with denoising stages, ManiFusion uses Trajectory-Fusion Scheduling (TFS), which dynamically adjusts fusion factors throughout sampling. ManiFusion generalizes across diffusion families, including both U-Net and transformer backbones in pixel and latent spaces. We further show that ManiFusion composes with existing acceleration methods and reaches throughput regimes unattainable by prior approaches with minimal quality degradation.

1 Introduction

Diffusion models have become the dominant framework for high-fidelity image generation by progressively refining noise into structured samples. [5, 30, 23] However, this strong generation quality comes at the cost of iterative denoising across many timesteps, making sampling inherently sequential and limiting throughput.[1] Prior work has made substantial progress by reducing the per-sample cost of diffusion sampling. One line of work reduces the number of denoising evaluations. [30, 31, 27, 12, 13, 38, 39, 40] Another line of work reduces the cost of individual denoising evaluations through feature reuse or caching. [14, 8] Together, these approaches have substantially improved the efficiency–quality trade-off of diffusion sampling in practice.

Despite these advances, further acceleration along the same per-sample axis is increasingly difficult. As denoising evaluations decrease, each remaining step must approximate more of the trajectory, making the dynamics more sensitive to approximation error. [34, 35] Caching methods reuse intermediate features across adjacent timesteps, but wider cache intervals amplify mismatch from fully recomputed trajectories. [14] When the two approaches are combined, reduced-step sampling makes each update more consequential while caching supplies features from increasingly mismatched trajectory states, limiting further gains without additional quality loss overall.

*Co-corresponding authors: Jangho Kim and Nojun Kwak.

[1] $\text{Throughput} = \frac{\text{images generated}}{\text{total inference time}} = \frac{\text{images per forward pass}}{\text{inference time per forward pass}}$.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

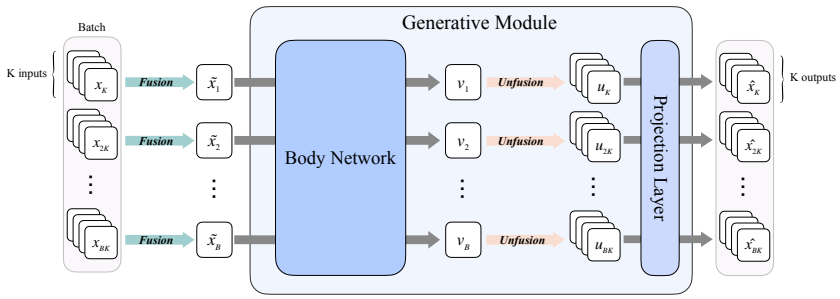


Figure 1: Illustration of the ManiFusion framework. $K \times B$ inputs is fused into B representations, jointly processed by a shared denoising backbone, then unfused via the *unfusion* module into K representations per group, and finally passed through a projection layer to produce K outputs.

As gains along the per-sample axis approach saturation, we instead increase throughput by processing more samples within each denoising evaluation. To this end, we propose *ManiFusion*, a lightweight framework for multi-sample generation through fused-space computation. ManiFusion processes multiple samples within a shared denoising backbone by fusing their representations before the backbone and separating them afterward through index-conditioned unfusion. Specifically, ManiFusion realizes multi-sample computation through *Trajectory-Fusion Scheduling* (TFS), which uses larger fusion factors in stable regions and smaller factors in sensitive stages.

However, generating multiple samples through a single fused representation presents a nontrivial challenge. Each sample needs a distinct denoising prediction, yet fusion combines their states into a representation the size of a single input. Naive superposition can entangle their information [37, 33] and compromise the denoising dynamics governing their individual generation trajectories. [30, 9] We examine how shared computation can preserve these distinct predictions through a geometric perspective under the manifold hypothesis. [20, 28, 1, 2] Low-dimensional data structure provides room for samples to coexist, while denoising requires their prediction targets to remain recoverable. Our analysis relates fusion-induced information loss to prediction error, supported by empirical studies of sample separation and denoising behavior.

To demonstrate that ManiFusion generalizes across diverse generative settings, we validate ManiFusion across major diffusion model families, including pixel-space U-Nets, diffusion transformers, and latent diffusion models. [24, 22, 23] Our experiments cover both unconditional and conditional generation across broad resolution ranges from CIFAR-10 to ImageNet. We further show that ManiFusion composes naturally with existing acceleration methods, including step-reduction samplers and caching-based acceleration. [12, 31, 14] By operating along a different throughput axis, ManiFusion reaches higher-throughput regimes beyond trajectory shortening or per-step acceleration alone.

2 Method

We propose ManiFusion, a backbone-agnostic framework for multi-sample generation through fused-space computation. Since multi-sample generation requires jointly processing multiple trajectories while preserving sample-wise prediction targets, we first establish ManiFusion through dual-space inference (Sec. 2.1). In this framework, multiple denoising evaluations are replaced by a single fused computation followed by index-conditioned recovery. We then extend ManiFusion across diverse diffusion backbones and sampling paradigms through Trajectory-Fusion Scheduling (TFS), which dynamically allocates the fusion factor according to trajectory dynamics (Sec. 2.2). Finally, we develop a geometric intuition for why shared computation can support distinct generation trajectories and what information fusion must preserve to make this possible (Sec. 2.3).

2.1 ManiFusion

ManiFusion enables multiple samples to share computation through fusion into a common representation and unfusion into individual outputs. We first formulate this principle for denoiser-level fusion. Diffusion models operate on an ambient representation space $\mathbb{R}^D$, corresponding to either pixel or

learned latent space depending on the model, with denoising (or flow) operator $f_t : \mathbb{R}^D \to \mathbb{R}^D$ at timestep t. Given K inputs $(x_t^1, \ldots, x_t^K) \in (\mathbb{R}^D)^K$, independent denoising aims to predict the sample-wise targets as $\big(f_t(x_t^1), \ldots, f_t(x_t^K)\big) \approx (y_t^1, \ldots, y_t^K)$. Both independent and fused models are trained against these targets y_t^i, with the fused model's predictions denoted by $\hat{y}_t^i$.

The key idea of ManiFusion is to replace these independent evaluations with a single computation in a shared fused space. To enable this, we first define a fusion operator $F_K : (\mathbb{R}^D)^K \to \mathbb{R}^D$ that combines representations through index-conditioned summation across sample slots

$$z_t = F_K(x_t^1, \ldots, x_t^K) = \sum_{i=1}^{K} \psi_{K,i}(x_t^i), \tag{1}$$

where $\psi_{K,i} : \mathbb{R}^D \to \mathbb{R}^D$ denotes the encoding function associated with the i-th sample slot.

Once the inputs are fused into a shared representation, computation proceeds through a fused-space operator $\tilde{f}_{t,K} : \mathbb{R}^D \to \mathbb{R}^D$, which is designed to approximate the effect of applying the independent operator f_t to each trajectory within the fused space. From this perspective, ManiFusion can be viewed as a dual-space computation framework, where generative dynamics are carried out through the coupled original sample space and fused shared space. After fused processing, individual predictions are recovered by the following index-conditioned unfusion operator

$$U_K(z)_i = \varphi_{K,i}(z), \quad i = 1, \ldots, K, \tag{2}$$

where $\varphi_{K,i} : \mathbb{R}^D \to \mathbb{R}^D$ denotes the decoding function for the i-th sample slot, including the final prediction head in this abstract notation. Thus, ManiFusion aims to satisfy

$$U_K\Big(\tilde{f}_{t,K}\left(F_K(x_t^1, \ldots, x_t^K)\right)\Big) \approx \big(f_t(x_t^1), \ldots, f_t(x_t^K)\big). \tag{3}$$

The following diagram illustrates how independent and fused computation aim to predict the same sample-wise targets.

$$\begin{array}{ccc} z_t & \xrightarrow{\quad\tilde{f}_{t,K}\quad} & \tilde{y}_t \\ \uparrow{\scriptstyle F_K} & & \downarrow{\scriptstyle U_K} \\ (x_t^1, \ldots, x_t^K) & \xrightarrow{\quad(f_t, \ldots, f_t)\quad} & (y_t^1, \ldots, y_t^K) \end{array} \tag{4}$$

where $z_t = F_K(x_t^1, \ldots, x_t^K)$ and $\tilde{y}_t = \tilde{f}_{t,K}(z_t)$.

Trajectory-Fusion Scheduling. For these denoiser-level variants, ManiFusion uses Trajectory-Fusion Scheduling (TFS), which allocates the fusion factor according to the local behavior of the generative trajectory. This is motivated by the observation that the approximation quality in Eq. 3 varies throughout the diffusion process, since the stability of dual-space computation depends on the local trajectory behavior. We therefore define the fusion factor $K_t = \kappa(t)$ as a function of timestep t. At each timestep, samples are grouped into tuples of size K_t, and fused computation is applied to each group as $z_t = F_{K_t}(x_t^1, \ldots, x_t^{K_t})$ and $(\hat{y}_t^1, \ldots, \hat{y}_t^{K_t}) = U_{K_t}(\tilde{f}_{t,K_t}(z_t))$. By allocating fusion factors according to trajectory dynamics, ManiFusion balances computational efficiency and sample fidelity across denoising stages without changing the backbone.

2.2 Instantiation Across Architectures and Generative Flows

To examine whether fused computation generalizes across model designs, we instantiate ManiFusion in U-Nets, diffusion transformers, and autoencoders, covering convolutional and attention-based backbones as well as denoiser-level and latent-representation sharing. We further consider DDPM, DDIM, and Flow Matching to study how this principle applies across different generative dynamics.

Backbone restructuring To maximize shared computation across multiple trajectories, ManiFusion restructures the denoiser such that most computation is performed in the fused space. Let $\mathcal{B}$ denote the heavy backbone body of the denoiser and $\mathcal{H}$ its final projection head. Instead of repeatedly evaluating $\mathcal{B}$ for each trajectory independently, ManiFusion places the fusion module F before $\mathcal{B}$ and

the unfusion module U after $\mathcal{B}$, allowing the dominant backbone computation to be shared across fused trajectories. This yields the following computational structure.

$$x_t^{(i,j)}{}_{j=1}^{K};\xrightarrow{F};\tilde{x}_t^{(i)};\xrightarrow{\mathcal{B}};\tilde{v}_t^{(i)};\xrightarrow{U(\cdot,j)};u_t^{(i,j)};\xrightarrow{\mathcal{H}};\hat{\epsilon}_t^{(i,j)}. \tag{5}$$

Here i indexes a fusion group and $j \in \{1, \ldots, K\}$ indexes a sample slot within that group.

MF-UNet For pixel-space models such as DDPM, we split the U-Net at its final convolutional layer, treating the multi-scale convolutional backbone as the shared body and the output convolution as the head. Fusion uses lightweight 3×3 convolutional blocks per index, while unfusion uses pointwise convolutions. Because the body output already encodes rich spatial features [24], simple channel-wise separation suffices without modifying the U-Net internals.

MF-DiT For transformer-based models such as DiT, inputs have shape $B \times K \times L \times d$, where L is the number of patch tokens. To treat the whole transformer stack as the shared body, we split the denoiser at the final linear projection. Fusion and unfusion are implemented as index-conditioned fully connected layers applied identically across patch positions $\ell \in \{1, \ldots, L\}$, sharing parameters to preserve token-wise structure and permutation sensitivity across all patch tokens.

MF-AE For latent diffusion models, fusion is applied at the representation level of the autoencoder rather than within the denoising trajectory. We extend ManiFusion by inserting fusion before encoding and unfusion after decoding, effectively operating on a shared latent representation. During autoencoder training, multiple inputs are fused prior to encoding and reconstructed through a shared latent. At generation time, a shared latent is denoised and decoded jointly. Unfusion is then applied at the output to recover individual samples. Because fusion operates at the representation level, K remains fixed during sampling.

Trajectory-dependent fusion allocation To show that trajectory-dependent fusion naturally generalizes across both score-based diffusion and velocity-based generative flows, we apply TFS by switching between independently trained fixed-K models during inference. For DDPM, larger fusion factors are assigned to early noisy stages and smaller factors to later stages, consistent with the coarse-to-fine behavior of diffusion trajectories [6, 35]. Specifically, we allocate 50% of timesteps to K=4, 25% to K=2, and 25% to K=1.[2] DDIM [30] uses the same allocation. For Flow Matching [9], we instead assign larger fusion factors to regions with smaller scheduler intervals, where the same velocity prediction error contributes a smaller displacement error to each update (Sec. 2.3). Finally, we combine ManiFusion with step-reduction samplers and per-step acceleration methods such as DeepCache [14], achieving additional throughput gains with only slight quality degradation.

2.3 Geometric Analysis

Fusing multiple samples reduces computation, but how can a single shared representation support the distinct predictions needed to generate each sample? To answer this question, we examine which differences between inputs must remain distinguishable after fusion for their predictions to be recovered. The low-dimensional structure of clean data provides a geometric basis for preserving these distinctions within a shared space. We then extend this reasoning to noisy inputs, where the information needed for prediction changes along the denoising trajectory, motivating adaptive allocation of shared computation.

Prediction information retained by fusion. For shared computation to support individual generation trajectories, fusion must preserve the information needed for the original sample-wise predictions. The training targets and squared-error objective remain unchanged, so the key question is whether these predictions can still be obtained through the shared representation. At timestep t, let $X_t = (x_t^1, \ldots, x_t^K)$ and $Y_t = (y_t^1, \ldots, y_t^K)$ denote the input and target tuples in $\mathbb{R}^{KD}$. Depending on the formulation, each target is noise [5], velocity [9], or a clean state [27]. Fusion maps the inputs to $Z_t = F_K(X_t) \in \mathbb{R}^D$, from which G_t produces the sample-wise predictions $\widehat{Y}_t = G_t(Z_t)$. Here G_t comprises the shared backbone, index-conditioned unfusion, and output head.

[2] This yields an idealized $2\times$ throughput since the effective cost is $\frac{1}{4} \times 0.5 + \frac{1}{2} \times 0.25 + 1 \times 0.25 = 0.5$.

To make this requirement precise, we use the optimal prediction from the full input tuple as a reference. Under squared loss, this is the conditional mean (Appendix C, Eq. 13)

$$m_t(X_t) := \mathbb{E}_t[Y_t \mid X_t],$$

where $\mathbb{E}_t$ denotes expectation at fixed t and available conditioning, such as class labels. If the shared representation retains the information needed to recover this prediction, there exists a predictor g_t such that $m_t(X_t) = g_t(Z_t)$ almost surely. Fusion can then attain the same minimum prediction error as the full-input reference. This condition identifies what fusion must preserve to share computation without introducing additional prediction error.

Prediction ambiguity under superposition. This criterion fails when tuples sharing a representation require different predictions. For a representation z, define its *fiber* as $F_K^{-1}(z) = \{X : F_K(X) = z\}$. If the optimal targets vary across this set, the fused predictor must compromise between them. The same squared-loss argument gives $g_t^*(Z_t) = \mathbb{E}_t[Y_t \mid Z_t] = \mathbb{E}_t[m_t(X_t) \mid Z_t]$ by iterated expectation, since $Z_t = F_K(X_t)$. The star denotes the fused optimum.

Define the minimum full-input risk as $R_{\text{full},t} = \mathbb{E}_t\|Y_t - m_t(X_t)\|^2$ and the minimum fused-input risk as $R_{F,t} = \mathbb{E}_t\|Y_t - g_t^*(Z_t)\|^2$. Their difference defines the *fusion ambiguity* as

$$\Delta_t(K) := R_{F,t} - R_{\text{full},t} = \mathbb{E}_t\|m_t(X_t) - g_t^*(Z_t)\|^2. \tag{6}$$

Thus collisions cause an irreducible prediction error when they hide differences in the required predictions. Agreement within each fiber makes this term vanish.

The learned predictor may also differ from g_t^*. Let θ denote its parameters and $R_{\theta,F,t} = \mathbb{E}_t\|Y_t - G_t(Z_t)\|^2$ its risk. Separating this prediction error from fusion ambiguity gives

$$R_{\theta,F,t} - R_{\text{full},t} = \underbrace{\Delta_t(K)}_{\text{information lost by fusion}} + \underbrace{\mathbb{E}_t\|g_t^*(Z_t) - G_t(Z_t)\|^2}_{\text{prediction error on retained information}}. \tag{7}$$

This decomposition separates the additional error into information lost through fusion and prediction error on the retained information. Appendix C, Eq. 17, derives both identities. Joint training can therefore reduce harmful collisions through F_K and improve recovery through G_t.

Manifold capacity and recoverable sample variations. To understand how several samples can fit in one representation, consider clean-state prediction, $Y_t = X_0$. At the noiseless endpoint $t = 0$, $m_0(X_0) = X_0$, so preserving the prediction means recovering the clean tuple X_0. Under the manifold hypothesis, each clean sample lies on a smooth d-dimensional manifold $\mathcal{M} \subset \mathbb{R}^D$ and needs only d independent local coordinates. A tuple of K independently varying samples lies on the product manifold $\mathcal{M}^K$, with Kd such coordinates.

Recovering these coordinates requires an ideal smooth decoder satisfying $g_0(F_K(X_0)) = X_0$. Let $T_{X_0}\mathcal{M}^K$ be the tangent space of allowed local sample variations, and $J_{\mathcal{M}}h$ the derivative of a map h restricted to these variations. Its rank counts the number of independent directions preserved by that map. Differentiating the recovery identity (Appendix C, Eq. 18) gives

$$Kd = \operatorname{rank}(J_{\mathcal{M}}m_0) \le \operatorname{rank}(J_{\mathcal{M}}F_K) \le D. \tag{8}$$

Thus smooth clean-tuple recovery requires $D \ge Kd$. This explains the room for sharing when $d \ll D$, since several samples' intrinsic coordinates can fit within one ambient representation. For MF-AE, the same argument uses D_{AE}, the dimension of its shared latent representation.

The capacity condition provides enough dimensions to represent a clean tuple, but preserving its information also requires separating the contributions of different slots. If changes in two slots map to the same direction, they can cancel under summation, preventing recovery even when $D \ge Kd$. This motivates the index-conditioned maps $\psi_{K,i}$, which allow each slot to encode its variations in different directions. Training against individual prediction targets encourages the fused representation to retain these distinctions. When the resulting directions are linearly independent on the clean manifold, the fusion map has full tangent rank and admits a local inverse (Appendix C). For recovery to be stable, these distinctions must also remain sufficiently large. The local bound $\|F_K(X) - F_K(X')\| \ge c\|X - X'\|$ with $c > 0$ quantifies this separation for nearby clean tuples, limiting inverse amplification to $1/c$. A small c therefore indicates sensitive recovery even when a local inverse exists. Together, capacity and local separation explain how a shared representation can support recoverable sample-wise information, motivating the toy, response, and training analyses below.

Table 1: Unified comparison of Vanilla, ManiFusion w.o. TFS, and ManiFusion across three sampling frameworks (DDPM, DDIM (T=50), FM) and DiT on CIFAR-10 and CelebA. ManiFusion w.o. TFS uses a fixed fusion factor with $K = 2$ throughout the denoising trajectory, while ManiFusion dynamically allocates the fusion factor via TFS. DiT is evaluated with class-conditional generation on CIFAR-10 only, while all other models are unconditional. Best results are in **bold**, and second-best results are underlined.

CIFAR-10

Method	Variant	Throughput↑	MAC↓	Parameter	FID↓	sFID↓	IS↑	Precision↑	Recall↑
DDPM	Vanilla	1.71	6.07T	35.7M	**3.41**	4.40	9.20	**0.68**	0.58
	Ours (-TFS)	3.17 (×1.85)	3.07T (50.6%)	+63.1K (0.18%)	5.93	4.58	8.67	0.63	0.58
	Ours	**3.31** (×1.94)	**3.06T** (50.4%)	+102.0K (0.29%)	3.88	**4.27**	**9.29**	0.66	**0.60**
DDIM (T=50)	Vanilla	35.47	303.44G	35.7M	**7.99**	6.66	**8.71**	**0.64**	0.55
	Ours (-TFS)	65.39 (×1.84)	153.44G (50.6%)	+63.1K (0.18%)	13.34	7.16	8.11	0.59	0.56
	Ours	**66.80** (×1.88)	**152.87G** (50.4%)	+102.0K (0.29%)	10.35	**6.29**	8.58	0.60	**0.57**
FM	Vanilla	9.60	124.55G	55.7M	**2.74**	**3.84**	**9.47**	**0.65**	**0.63**
	Ours (-TFS)	18.22 (×1.90)	62.89G (50.5%)	+185.9K (0.33%)	6.72	4.50	8.76	0.62	0.60
	Ours	**18.29** (×1.91)	**62.70G** (50.3%)	+323.4K (0.58%)	4.10	3.95	8.72	0.61	0.59
DiT	Vanilla	8.25	217.00G	32.48M	**10.58**	7.37	**10.17**	**0.75**	**0.40**
	Ours (-TFS)	15.86 (×1.92)	114.25G (52.6%)	+1.04M (3.02%)	11.13	**6.45**	9.50	0.72	**0.40**
	Ours	**15.95** (×1.93)	**113.84G** (52.5%)	+1.63M (5.02%)	10.97	7.71	9.60	0.73	0.37

CelebA

Method	Variant	Throughput↑	MAC↓	Parameter	FID↓	sFID↓	IS↑	Precision↑	Recall↑
DDPM	Vanilla	0.71	15.56T	113.7M	**3.51**	6.27	**2.77**	**0.76**	0.50
	Ours (-TFS)	1.28 (×1.80)	7.92T (50.9%)	+63.1K (0.06%)	4.73	6.98	2.70	0.75	0.46
	Ours	**1.33** (×1.87)	**7.87T** (50.6%)	+102.0K (0.09%)	3.81	**6.00**	**2.77**	0.74	**0.51**
DDIM (T=50)	Vanilla	14.56	778.00G	113.7M	**5.58**	8.37	2.81	**0.74**	0.46
	Ours (-TFS)	26.05 (×1.79)	395.88G (51.1%)	+63.1K (0.06%)	7.43	9.07	2.75	0.71	0.42
	Ours	**26.83** (×1.84)	**393.63G** (50.6%)	+102.0K (0.09%)	6.52	**7.82**	**2.84**	0.68	**0.48**
FM	Vanilla	2.12	589.55G	169.1M	**2.80**	5.35	**3.23**	0.67	**0.60**
	Ours (-TFS)	4.06 (×1.92)	297.24G (50.4%)	+185.9K (0.11%)	3.67	5.67	3.17	**0.68**	0.56
	Ours	**4.08** (×1.92)	**296.49G** (50.3%)	+323.4K (0.19%)	2.98	**5.20**	3.22	0.66	0.59

Noisy targets and trajectory-dependent allocation. The clean-manifold argument concerns the Kd intrinsic coordinates of a clean tuple. With nonzero noise scale, independent Gaussian noise introduces variation in all D ambient directions of each sample, giving the noisy tuple full support in $\mathbb{R}^{KD}$. Thus, for $K > 1$, continuous fusion into $\mathbb{R}^D$ cannot distinguish every noisy tuple (Appendix C). We therefore return to target agreement within fibers, quantified by Eq. 6. For velocity targets, m_t is the conditional mean velocity of each sample. Small variation of this field within a fiber is precisely the vector-field compatibility needed for shared prediction.

To understand how noise affects this agreement, we must account for the prediction target. Write the forward-noised tuple as $X_t = a_t X_0 + \sigma_t \epsilon$, where a_t scales the clean signal, $\sigma_t > 0$ scales the noise, and ϵ is the Gaussian noise tuple. For noise prediction, rearranging $\epsilon = (X_t - a_t X_0)/\sigma_t$ and conditioning on X_t yields

$$m_t^{\epsilon}(X_t) = \frac{X_t - a_t \mathbb{E}_t[X_0 \mid X_t]}{\sigma_t}. \tag{9}$$

Here m_t^{ϵ} denotes the optimal noise prediction. Even at the pure-noise limit $a_t = 0, \sigma_t = 1$, it equals X_t, whereas clean-state prediction uses the posterior mean of X_0. Thus noise level and target parameterization together determine the information required by fusion.

Choosing when to share more computation also requires knowing how prediction errors affect sampling. For a velocity model, let t' be the next sampling time, $h_t = t' - t$, $\widehat{v}_t = G_t(F_K(X_t))$, and $v_t^* = m_t(X_t)$. Subtracting the Euler updates from the same state, $\widehat{X}_{t'} = X_t + h_t \widehat{v}_t$ and $X_{t'}^* = X_t + h_t v_t^*$, gives

$$\|\widehat{X}_{t'} - X_{t'}^*\| = |h_t| \, \|\widehat{v}_t - v_t^*\|. \tag{10}$$

For a fixed velocity discrepancy, a smaller interval reduces its contribution to that update. This motivates larger fusion factors at smaller Flow Matching intervals. In DDPM/DDIM, coarse-to-fine generation motivates sharing more computation early and restoring sample-specific capacity during refinement. TFS applies these design principles, and its schedule ablations evaluate the resulting quality–throughput trade-off.

CIFAR-10

Throughput	Variant	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
66.80 img/s	Vanilla	10.36	8.16	8.54	**0.63**	0.53
	Ours	**10.35**	**6.29**	**8.58**	0.60	**0.57**
156.56 img/s	Vanilla	17.55	12.66	8.15	**0.59**	0.46
	Ours	**14.30**	**8.76**	**8.32**	0.58	**0.53**
317.89 img/s	Vanilla	43.59	26.38	6.83	0.50	0.30
	Ours	**22.99**	**13.76**	**7.84**	**0.55**	**0.46**
565.03 img/s	Vanilla	99.09	51.83	4.80	0.38	0.15
	Ours	**50.07**	**28.11**	**6.48**	**0.48**	**0.31**

CelebA

Throughput	Variant	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
26.83 img/s	Vanilla	7.12	10.68	2.77	**0.73**	0.43
	Ours	**6.52**	**7.82**	**2.84**	0.68	**0.48**
66.90 img/s	Vanilla	12.33	19.01	2.59	**0.73**	0.30
	Ours	**8.90**	**11.53**	**2.72**	0.69	**0.40**
127.92 img/s	Vanilla	23.06	31.96	2.37	0.68	0.15
	Ours	**14.11**	**18.63**	**2.53**	**0.69**	**0.28**
228.77 img/s	Vanilla	65.15	68.64	2.13	0.40	0.01
	Ours	**31.24**	**36.63**	**2.26**	**0.60**	**0.10**

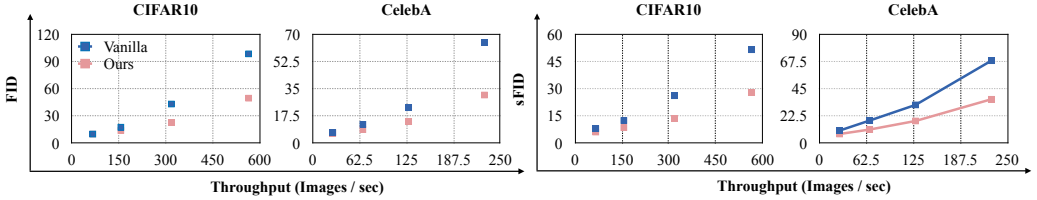


Figure 2: Comparison of throughput vs. quality and diversity on CIFAR-10 (left) and CelebA (right). The table (top) reports FID, sFID, IS, Precision, and Recall across throughput levels for Vanilla and Ours). The plots (bottom) visualize FID (right) and sFID (left) for DDIM, highlighting how Ours (red) compares against Vanilla DDIM (blue).

3 Experiments

Diffusion Models and Baselines. Our goal is to validate ManiFusion as a general computational restructuring principle that operates across diverse generative settings, rather than a technique specialized to a particular pretrained foundation model stack. We therefore evaluate ManiFusion across major diffusion model families spanning different denoising spaces, backbone architectures, and conditional generation settings. For pixel-space U-Nets, we apply *MF-UNet* to DDPM [5] on CIFAR-10 and CelebA. For pixel-space transformers and conditional generation, we evaluate *MF-DiT* on class-conditional DiT [22] on CIFAR-10. For latent-space autoencoders, we apply *MF-AE* to LDM [23] on CelebA-HQ. Finally, for latent-space transformers and conditional generation, we evaluate *MF-DiT* on ImageNet using DiT [22]. These experiments demonstrate that ManiFusion generalizes across diffusion model families rather than specific architectures, scaling across resolutions, denoising spaces, and conditioning regimes.

Experimental settings. All throughput measurements are conducted using batched inference for vanilla baselines and ManiFusion under identical inference environments, including sufficiently large batch configurations, scheduler settings, and hardware setup, to ensure fair comparison. Throughput is reported using end-to-end wall-clock generation time, and computational overhead is evaluated using parameter count and multiply–accumulate operations (MACs). Unless otherwise specified, all ManiFusion results use fusion factors up to K=4, and all models are trained from scratch without finetuning or freezing. Additional details and ablations appear in the appendix.

3.1 Application to Baseline Models

We first evaluate ManiFusion on baseline diffusion models, where it achieves nearly 2× throughput while maintaining sample quality close to the baseline without materially increasing model complexity(Tab. 1). Qualitative results in the Appendix confirm preserved perceptual fidelity and diversity. We observe similar behavior in transformer-based diffusion models. Interestingly, mixing different class conditions within fused groups is more stable than grouping identical classes, as verified by additional experiments in the Appendix. This supports the interpretation that conditional diversity provides cues for slot separation (Appendix C).

We extend ManiFusion to latent diffusion in two ways. On CelebA-HQ, we apply *MF-AE* to LDM(Tab. 3). We also implement MF-UNet directly on the LDM backbone, with results provided in the Appendix. Empirically, MF-AE exhibits smaller quality degradation than MF-UNet at the same fusion level. This comparison highlights the role of where the shared bottleneck is introduced.

Table 2: Combination of ManiFusion w.o. TFS and ManiFusion with DeepCache on CIFAR-10 and CelebA. ManiFusion w.o. TFS uses a fixed fusion factor with $K = 2$, while ManiFusion dynamically allocates the fusion factor via TFS. MAC reduction ratios and throughput speedups are shown in parentheses. *DDIM steps are reduced to match the same throughput setting.

CIFAR-10								
Setting	Variant	Throughput↑	MAC↓ (G)	FID↓	sFID↓	IS↑	Prec↑	Rec↑
DDIM	Vanilla	17.70	606.88	6.30	5.65	8.88	0.65	0.56
	+DeepCache	28.05 (×1.59)	390.18 (64.3%)	6.90	6.21	8.89	0.64	0.56
	+Step-Reduction	49.02* (×2.77)	207.88 (34.3%)	10.12	7.40	8.56	0.63	0.53
Ours (-TFS)	Vanilla	33.48 (×1.89)	306.88 (50.6%)	11.21	6.12	8.30	0.59	0.57
	+DeepCache	51.19 (×2.89)	198.48 (32.7%)	11.24	6.48	8.29	0.59	0.56
Ours	Vanilla	32.90 (×1.86)	307.76 (50.7%)	8.65	5.46	8.68	0.61	0.59
	+DeepCache	51.98* (×2.94)	197.38 (32.5%)	9.42	6.37	8.62	0.61	0.57
CelebA								
Setting	Variant	Throughput↑	MAC↓ (G)	FID↓	sFID↓	IS↑	Prec↑	Rec↑
DDIM	Vanilla	7.20	1556.01	4.17	6.70	2.82	0.74	0.50
	+DeepCache	9.81 (×1.36)	1075.81 (69.1%)	6.07	9.10	2.79	0.72	0.45
	+Step-Reduction	16.83* (×2.34)	572.58 (36.8%)	8.22	11.38	2.73	0.70	0.40
Ours (-TFS)	Vanilla	13.25 (×1.84)	791.76 (50.9%)	6.17	7.71	2.76	0.71	0.45
	+DeepCache	16.96 (×2.36)	551.66 (35.5%)	8.43	10.50	2.71	0.69	0.40
Ours	Vanilla	13.41 (×1.86)	787.26 (50.6%)	5.54	6.91	2.85	0.69	0.51
	+DeepCache	17.65* (×2.45)	547.18 (35.2%)	7.07	8.69	2.79	0.68	0.46

Table 3: Performance of ManiFusion in latent diffusion settings. MF-AE is evaluated on CelebA-HQ using LDM, while MF-DiT is evaluated on ImageNet using DiT-S/2 with fusion factors $K = 2, 4$. For comparison with throughput gains from larger patch sizes, we also report DiT-S/4 results.

CelebA-HQ								
Method	Throughput↑	MAC↓ (G)	Parameter↓	FID↓	sFID↓	IS↑	Prec↑	Rec↑
LDM (baseline)	0.54	96.38T	362.3M	11.58	12.68	3.12	0.68	0.34
MF-AE (K=2)	1.08 (×2.00)	48.19T (50.0%)	+159.8K (0.04%)	12.90	13.74	2.96	0.66	0.33
MF-AE (K=4)	2.16 (×4.00)	24.09T (25.0%)	+319.6K (0.08%)	14.57	16.79	2.78	0.69	0.24
ImageNet								
Method	Throughput↑	MAC↓ (G)	Parameter↓	FID↓	sFID↓	IS↑	Prec↑	Rec↑
DiT-S/2 (baseline)	3.04	1067.50G	32.9M	71.85	12.29	18.65	0.35	0.55
Ours (-TFS)	5.95 (×1.96)	562.50G (52.69%)	+1.33M (4.04%)	78.22	15.38	16.83	0.34	0.53
Ours	5.98 (×1.97)	551.95G (51.70%)	+1.94M(5.90%)	74.19	14.85	19.08	0.34	0.49

MF-AE learns a representation for joint reconstruction, whereas MF-UNet shares the prediction computation at each denoising step. We further evaluate *MF-DiT* on ImageNet for large-scale class-conditional generation. These results show strong quality across latent U-Net and transformer backbones in both settings.

3.2 Orthogonal Acceleration Beyond Existing Throughput Limits

We extend ManiFusion to step-reduction frameworks, demonstrating that per-step multi-sample acceleration is complementary to trajectory shortening. As shown in Fig. 2, ManiFusion maintains higher sample quality at matched throughput. In contrast, baseline methods rapidly degrade beyond their practical throughput ceiling, highlighting the advantage of trajectory-dependent fusion allocation in high-throughput regimes.

Finally, we combine ManiFusion with DeepCache to demonstrate compatibility with caching-based acceleration. At matched throughput, the combined approach consistently outperforms configurations relying solely on step-reduction and caching(Tab. 2). These results suggest that compressing computation along a single axis is fundamentally limited, whereas combining orthogonal acceleration strategies yields compounding efficiency gains in high-throughput settings.

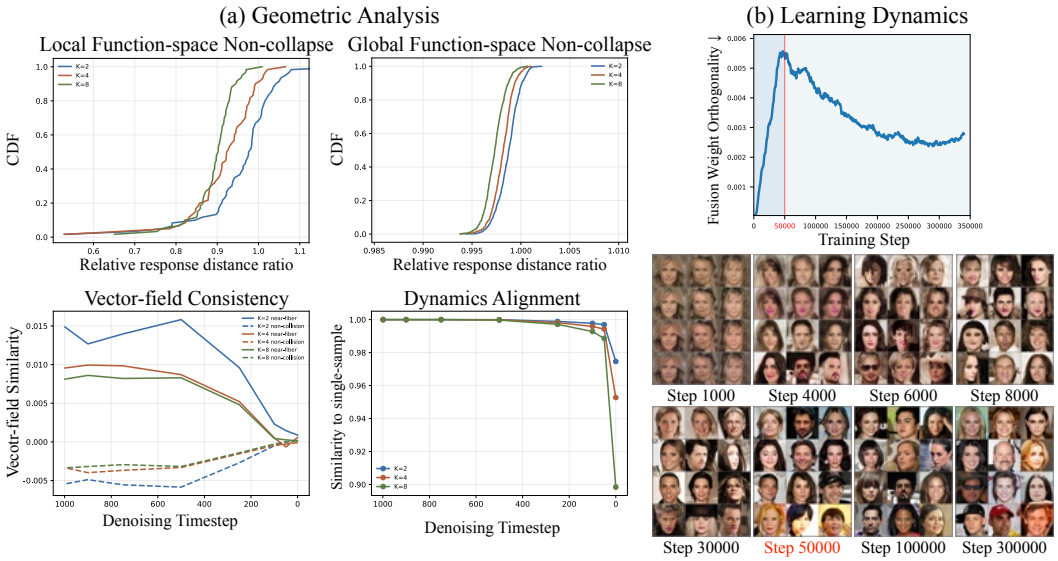


Figure 3: Analysis of the representation geometry induced by ManiFusion. (a) Geometric diagnostics measure prediction separation using response-distance ratios, and vector-field consistency and dynamics alignment using cosine similarity. (b) Training dynamics of Fusion Weight Orthogonality for MF-AE with $K = 4$, showing the development of slot-specific separation during training.

4 Analysis

We analyze the geometry and dynamics induced by multiplexed generation under ManiFusion. In particular, we study whether fusion preserves distinguishability, maintains consistent generative dynamics, and develops stable separation during training. To this end, we examine prediction separation, vector-field consistency, dynamics alignment, and fusion-module behavior. Additional visualizations and toy analysis on the $D \geq Kd$ condition are provided in the appendix.

4.1 Empirical Validation of the Geometric Analysis

To examine whether fusion retains prediction-relevant distinctions (Sec. 2.3), we compare velocity-prediction distances with those of the single-sample model. Figure 3(a) reports their ratios for same-class local perturbations and random tuple pairs. Ratios near one indicate comparable prediction separation. The random-pair ratios concentrate near one, whereas local ratios shift downward as K increases. Fusion thus preserves broad prediction differences while attenuating finer distinctions under stronger compression. The Appendix further examines local sensitivity.

Sharing a representation also requires compatible predictions. To examine this requirement, we compare slot-wise vector-field directions between complete tuples. Collision pairs have nearby fused representations, approximating a shared fiber, while non-collision pairs are farther apart. Near-fiber pairs exhibit higher cosine similarity over much of the trajectory, indicating a relative tendency toward more compatible field directions. To connect this agreement to generation, we also assess alignment with the reference dynamics. Alignment with the single-sample model remains close to one in early stages but declines near the end, particularly for $K = 8$. These results motivate TFS, which shares more computation early and restores sample-specific capacity during final refinement. Schedule ablations evaluate the quality–throughput trade-off.

4.2 Training Dynamics of the Fusion Module

To further understand the fusion mechanism, we analyze the training dynamics of the fusion module through *Fusion Weight Orthogonality*, defined as the average squared inner product between normalized fusion weight vectors. Figure 3 shows that the value initially increases and then decreases after approximately 50k steps. Early in training, different streams produce nearly identical faces, and separation emerges progressively as individual streams diverge in structure. Once the generated

images begin to exhibit distinct color patterns across streams, Fusion Weight Orthogonality starts to decrease, indicating that the fusion weight vectors become increasingly orthogonal. This suggests that the fusion module gradually reorganizes its parameter space to allocate independent slot-specific directions for different samples, supporting the trajectory-wise separation discussed in Sec. 2.3.

5 Related Work

Diffusion Models Diffusion models have become a dominant class of generative models by learning to iteratively denoise corrupted signals [29, 5, 32]. Pixel-space approaches such as DDPM operate directly in image space [5], while Latent Diffusion Models (LDMs) perform denoising in compressed but still high-capacity latent space [23]. Recent transformer-based variants such as DiT [22] demonstrate that larger latent representations and L/XL-scale backbones improve performance but also increase inference cost. Across these formulations, diffusion models leave substantial representational slack for ManiFusion, increasing the potential of fused multi-sample computation at scale.

Diffusion Model Acceleration The primary bottleneck of diffusion models is iterative inference, where generating a single sample requires many sequential denoising steps and repeated network evaluations. A large body of work therefore focuses on reducing the number of sampling steps. For example, DDIM reformulates the reverse process with fewer iterations [30, 18], while flow-matching methods learn continuous-time vector fields that directly transport noise to data [9, 10]. Another line of work reduces the cost of each sampling step through caching. Methods such as DeepCache and CacheMe reuse intermediate activations across timesteps to avoid redundant computation and lower per-step latency. [19, 34, 36] However, these approaches still process each trajectory independently and therefore do not improve multi-sample throughput within a single denoising evaluation.

Superposition in Deep Learning A line of work has explored accelerating neural networks by superposing multiple inputs into a shared representation. For example, the Byte Latent Transformer [21] leverages sparsity to process groups of tokens jointly, while Latent Attention [11] fuses multiple tokens into a shared latent representation for attention. Liu.al. [16] extended input superposition to sentence- and token-level classification, while Menet et al. [15] applied the concept to the classification of small-scale image datasets such as MNIST and CIFAR-10. In contrast, we are the first to apply superposition to generative modeling by fusing multiple signals within a latent trajectory, allowing one inference process to generate multiple images without quality loss.

6 Conclusion

In this paper, we proposed ManiFusion, a diffusion acceleration framework that improves throughput by processing multiple samples within a single denoising evaluation. Rather than relying only on trajectory shortening or per-step latency reduction, ManiFusion introduces a complementary throughput axis by increasing the number of samples processed per evaluation. Our geometric analysis explains how shared representations can retain sample-wise prediction information, while trajectory-aware fusion allocation balances throughput and generation quality. Across diverse diffusion settings, ManiFusion operates on top of existing acceleration strategies and enables throughput regimes difficult to achieve with them alone. These results position ManiFusion as a scalable direction for future diffusion acceleration.

Limitations A limitation of ManiFusion is that it currently requires full pretraining to integrate fused computation into the denoising backbone, resulting in substantial computational cost and preventing lightweight finetuning-based adaptation. In addition, we do not evaluate large-scale text-to-image foundation models, limiting our study to discussing potential scalability rather than empirically validating such settings. Finally, performance can still degrade under extreme fusion factors, suggesting remaining limits in high-throughput scaling.

Acknowledgements

This work was supported by the Korean Government through the grants from IITP (RS-2021-II211343, RS-2022-II220320, RS-2025-25442338) and Samsung Electronics Co., Ltd (IO251219-14870-01).

References

[1] Gunnar Carlsson. Topology and data. *Bulletin of the American mathematical society*, 46(2):255–308, 2009.

[2] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

[6] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[7] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

[8] Senmao Li, Taihang Hu, Joost Van De Weijer, Fahad S Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of the encoder for diffusion model inference. *Advances in neural information processing systems*, 37:85203–85240, 2024.

[9] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[10] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[11] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[12] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

[13] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, 2025.

[14] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15762–15772. IEEE, 2024.

[15] Nicolas Menet, Michael Hersche, Geethan Karunaratne, Luca Benini, Abu Sebastian, and Abbas Rahimi. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. *Advances in Neural Information Processing Systems*, 36:39553–39565, 2023.

[16] Vishvak Murahari, Carlos Jimenez, Runzhe Yang, and Karthik Narasimhan. Datamux: Data multiplexing for neural networks. *Advances in Neural Information Processing Systems*, 35:17515–17527, 2022.

[17] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

[18] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PmLR, 2021.

[19] Mehdi Noroozi, Isma Hadji, Brais Martinez, Adrian Bulat, and Georgios Tzimiropoulos. You only need one step: Fast super-resolution with stable diffusion via scale distillation. In *European Conference on Computer Vision*, pages 145–161. Springer, 2024.

[20] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[21] Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, 2025.

[22] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182. IEEE, 2023.

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 10674–10685. ieee, 2022.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[25] MS Sajjadi, O Bachem, M Lucic, O Bousquet, and S Gelly. Assessing generative models via precision and recall. arxiv preprint. *arXiv preprint arXiv:1806.00035*, 2018.

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[27] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[28] Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

[29] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[31] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[32] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[33] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pages 6438–6447. PMLR, 2019.

[34] Mengfei Xia, Yujun Shen, Changsong Lei, Yu Zhou, Ran Yi, Deli Zhao, Wenping Wang, and Yong-Jin Liu. Towards more accurate diffusion model acceleration with a timestep tuner, 2025.

[35] Shuai Yang, Yukang Chen, Luozhou Wang, Shu Liu, and Yingcong Chen. Denoising diffusion step-aware models, 2024.

[36] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Resshift: Efficient diffusion model for image super-resolution by residual shifting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[37] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[38] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

[39] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7777–7786. IEEE, 2024.

[40] Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. *Advances in Neural Information Processing Systems*, 37:40831–40860, 2024.

A Appendix Overview

This appendix follows the main paper from implementation to analysis and evaluation. We first provide training and inference configurations, followed by derivations supporting the geometric analysis. We then examine clean-manifold capacity, local sensitivity, interpolation, and feature-space coverage empirically. Next come extensions through larger fusion factors, TFS, DeepCache, and latent diffusion, followed by conditional grouping and kernel-size ablations. We conclude with throughput-oriented baseline comparisons and qualitative results organized by sampler, throughput, and latent fusion design.

B Implementation Details

We describe here the training and inference configurations and metric informations used across all datasets and model families.

Metrics We assess perceptual quality and distributional coverage along five complementary axes. First, Inception Score (IS) [26] and Fréchet Inception Distance (FID) [4] to jointly capture sample diversity and global visual fidelity. Second, we compute the spatial FID (sFID), which measures the Fréchet distance over intermediate feature maps that preserve spatial structure, providing a more sensitive evaluation of local detail and geometric fidelity [17, 3]. We additionally report precision and recall [7, 25] to quantify manifold fidelity and coverage under fusion. Moreover, all Macs, Throughputs are measured by a single NVIDIA A6000 GPU with batched inference of batch 128.

Configurations For CIFAR-10 DDPM, we follow the protocol of Ho et al. [5], while CelebA DDPM and the DDIM samplers adopt the settings of Song et al. [30]. For Flow Matching on CIFAR-10, we replicate Lipman et al. [10]. For CelebA, no official baseline exists, so we define its configuration consistently with the rest of our experiments. For latent diffusion models, we follow the training and sampling setup of Rombach et al. [23]. For transformer-based diffusion models, we adopt the DiT configuration proposed by Peebles and Xie [22] for ImageNet.

Across all experiments, models are trained with a fixed fusion factor K. Each model is trained independently for its corresponding K, and no mixture of fusion factors is used during training. The index embeddings used for demultiplexing are therefore defined only for the corresponding K and remain fixed throughout training and inference. We do not introduce additional regularization or curriculum strategies. Separation emerges from the architecture and training objective.

For Trajectory-Fusion Scheduling(TFS), we combine models trained with different fusion factors during inference. The denoising trajectory is partitioned into intervals and each interval uses the model trained with the corresponding K. This allows large fusion factors in early timesteps for higher throughput while gradually reducing K later to preserve generation quality.

For the denoiser-level variants, TFS switches among independently trained fixed-K predictors while the sampler retains the individual sample states. Each selected predictor supplies slot-wise update targets as in Eq. 12. Changing K therefore changes the amount of shared prediction computation while preserving the sample-wise sampling interface. We use trajectory stage for DDPM/DDIM and scheduler intervals for Flow Matching to allocate this computation. The latter choice follows the step-size scaling in Eq. 23. Schedule ablations measure the resulting quality–throughput trade-off. MF-AE uses a fixed K throughout shared-latent generation and unfuses after decoding.

Across all cases, the same learning rate schedule, noise schedule, and sampling procedure are maintained for each fixed-K model. Evaluation is performed on 50k generated samples using the implementation of Dhariwal et al. [3], ensuring that any performance differences can be attributed solely to the ManiFusion modules. The detailed configuration is provided in Table 4.

Table 4: Training and inference configurations of DDPM, DDIM, and Flow Matching on CIFAR-10 and CelebA, including model architecture parameters, diffusion sampling schedules, and optimization hyperparameters.

	CIFAR-10			**CelebA**		
	DDPM	DDIM	Flow Matching	DDPM	DDIM	Flow Matching
Channels	128	128	128	128	128	128
Channel Multiplier	1,2,2,2	1,2,2,2	2,2,2	1,1,2,2,4,4	1,1,2,2,4,4	2,2,4,4
Number of Heads	1	1	1	16,16,32,32,64,64	16,16,32,32,64,64	1
Activation Function	SiLU	SiLU	SiLU	SiLU	SiLU	SiLU
Diffusion Steps	1000	50	50	1000	50	50
Noise Schedule	linear	linear	–	linear	linear	–
Beta Start	0.0001	0.0001	–	0.0001	0.0001	–
Beta End	0.02	0.02	–	0.02	0.02	–
Dropout	0.1	0.1	0.3	0.0	0.0	0.3
Learning Rate	2e-4	2e-4	1e-4	2e-4	2e-4	1e-4
Batch Size	128	128	64	32	32	64

C Derivations for the Geometric Analysis

We analyze fusion through the information required for prediction. The central principle is that complete tuples mapped to the same fused representation should require compatible prediction targets. Clean-manifold capacity, fiber-wise consistency at noisy timesteps, and trajectory-dependent fusion allocation are different aspects of this principle. The timestep-wise analysis concerns MF-UNet and MF-DiT. The clean-reconstruction argument also describes the representation bottleneck of MF-AE. The risk decomposition establishes the prediction criterion. The geometric arguments explain how representation capacity and separation support it. The experiments then examine these mechanisms in learned models.

General targets and harmful fusion collisions. At timestep t, let $X_t = (x_t^1, \dots, x_t^K)$ denote the input tuple and $Y_t = (y_t^1, \dots, y_t^K)$ its training targets. A target may be noise, velocity, a score-related quantity, a flow-matching target, or a clean state. Using uppercase letters for the complete tuples in this analysis, the computation in Eq. 3 becomes

$$Z_t = F_K(X_t), \qquad \widehat{Y}_t = G_t(Z_t), \qquad m_t(X_t) := \mathbb{E}[Y_t \mid X_t, t]. \tag{11}$$

Here G_t comprises the backbone, index-conditioned unfusion, and output head. Its dependence on K is implicit. The sampler combines this prediction with the individual states as described below. Any additional conditioning available to the predictor is held fixed or included in the conditioning of the expectations below. The relevant requirement is task-relevant target identifiability. Ideally, $m_t(X_t) = g_t(F_K(X_t))$ almost surely for some fused-space function g_t. Distinct tuples can therefore share a representation without loss when their optimal targets agree. For update targets, the sampler already receives the current states, giving

$$\widehat{X}_{t'} = S_{t\to t'}(X_t, G_t(F_K(X_t)))\,, \tag{12}$$

where t' is the next sampling time and $S_{t\to t'}$ denotes the update (with sampler randomness suppressed). The computation therefore has two complementary roles. The predictor G_t predicts the sample-wise targets from the shared representation, and $S_{t\to t'}$ applies them to the individual states. The target parameterization determines which information the fused branch must retain.

Conditional means and the risk identities. We make explicit the steps leading to Eqs. 6 and 7. Fix t, the available conditioning, and a deterministic measurable fusion map F_K. Assume $\mathbb{E}_t\|Y_t\|^2 < \infty$ and consider square-integrable predictors. Here $\mathbb{E}_t$ uses the same fixed conditioning as in the main text. Write $m = m_t(X_t)$ and let $q = q(X_t)$ be any full-input predictor. Since $\mathbb{E}_t[Y_t - m \mid X_t] = 0$, iterated expectation gives $\mathbb{E}_t\langle Y_t - m, m - q\rangle = 0$, where $\langle\cdot,\cdot\rangle$ is the Euclidean inner product. Expanding $Y_t - q = (Y_t - m) + (m - q)$ therefore yields

$$\mathbb{E}_t\|Y_t - q\|^2 = \mathbb{E}_t\|Y_t - m\|^2 + \mathbb{E}_t\|m - q\|^2. \tag{13}$$

The second term is nonnegative and vanishes at $q = m$. This proves that the conditional mean is the optimal full-input prediction, uniquely up to almost-sure equality. Applying the same argument

to predictors observing Z_t gives the optimum $\mathbb{E}_t[Y_t \mid Z_t]$. Because Z_t is a function of X_t, iterated expectation identifies this optimum as $g_t^*(Z_t) = \mathbb{E}_t[m_t(X_t) \mid Z_t]$.

Under squared loss, define the population risks

$$\begin{aligned} R_{\mathrm{full},t} &= \inf_q \mathbb{E}[\|Y_t - q(X_t)\|^2 \mid t], \\ R_{F,t} &= \inf_g \mathbb{E}[\|Y_t - g(F_K(X_t))\|^2 \mid t], \end{aligned} \tag{14}$$

where the infima range over all square-integrable predictors with the indicated inputs, for a fixed fusion map. The optimal fused predictor is $g_t^*(z) = \mathbb{E}[m_t(X_t) \mid F_K(X_t) = z, t]$, giving

$$\begin{aligned} \Delta_t(K) &:= R_{F,t} - R_{\mathrm{full},t} \\ &= \mathbb{E}\Big[\|m_t(X_t) - \mathbb{E}[m_t(X_t) \mid F_K(X_t), t]\|^2 \mid t\Big]. \end{aligned} \tag{15}$$

This identity applies to any square-integrable target Y_t. The excess risk is the expected conditional variance of the optimal target within each fiber. It vanishes exactly when $m_t(X_t)$ is determined by Z_t almost surely. Thus, overlap between input representations is compatible with accurate prediction when the corresponding targets agree. For the finite learned predictor, let $R_{\theta,F,t} = \mathbb{E}[\|Y_t - G_t(F_K(X_t))\|^2 \mid t]$. Its excess risk decomposes as

$$\begin{aligned} R_{\theta,F,t} - R_{\mathrm{full},t} = &\underbrace{\mathbb{E}[\|m_t(X_t) - g_t^*(Z_t)\|^2 \mid t]}_{\text{irreducible fusion ambiguity } \Delta_t(K)} \\ &+ \underbrace{\mathbb{E}[\|g_t^*(Z_t) - G_t(Z_t)\|^2 \mid t]}_{\text{approximation and optimization error}}. \end{aligned} \tag{16}$$

To verify the decomposition directly, abbreviate $g^* = g_t^*(Z_t)$ and $G = G_t(Z_t)$ and split the error into three residuals,

$$Y_t - G = \underbrace{Y_t - m}_{A} + \underbrace{m - g^*}_{B} + \underbrace{g^* - G}_{C}. \tag{17}$$

The residual A has zero conditional mean given X_t, and B has zero conditional mean given Z_t. Since B and C are functions of X_t, and C is also a function of Z_t, their cross terms vanish,

$$\begin{aligned} \mathbb{E}_t\langle A, B\rangle &= \mathbb{E}_t\langle \mathbb{E}_t[A \mid X_t], B\rangle = 0, \\ \mathbb{E}_t\langle A, C\rangle &= \mathbb{E}_t\langle \mathbb{E}_t[A \mid X_t], C\rangle = 0, \\ \mathbb{E}_t\langle B, C\rangle &= \mathbb{E}_t\langle \mathbb{E}_t[B \mid Z_t], C\rangle = 0. \end{aligned}$$

Consequently, $R_{\theta,F,t} = \mathbb{E}_t\|A\|^2 + \mathbb{E}_t\|B\|^2 + \mathbb{E}_t\|C\|^2$. The first term is $R_{\mathrm{full},t}$. Setting $G = g^*$ makes $C = 0$ and gives $R_{F,t} - R_{\mathrm{full},t} = \mathbb{E}_t\|B\|^2$, proving Eq. 6. Restoring the actual G adds $\mathbb{E}_t\|C\|^2$, proving Eq. 7. Zero ambiguity is equivalent to $m = g^*$ almost surely, so the information criterion concerns fibers represented in the data distribution. This decomposition explains the learning pressure on the fusion modules. For a fixed F_K, the predictor approaches the conditional mean of the retained targets. Updating F_K also changes the fibers and hence the ambiguity term. End-to-end training can therefore improve both the organization of target information and its recovery, linking the regression objective to the emergence of slot separation.

Clean-state prediction as an interpretable special case. For $Y_t = X_0$, the optimal target is $m_t(X_t) = \mathbb{E}[X_0 \mid X_t, t]$. Near the low-noise endpoint, $m_t(X_t) \approx X_0$, so target identifiability approximately becomes clean-tuple identifiability. Suppose each clean sample lies on a smooth d-dimensional manifold $\mathcal{M} \subset \mathbb{R}^D$. A tuple lies on $\mathcal{M}^K$, of dimension Kd, while the fused representation has dimension D. Writing $t = 0$ for the clean endpoint in this limiting argument, $m_0(X_0) = X_0$. Suppose smooth exact recovery holds on a local patch of the clean manifold, so that $g_0 \circ F_K$ equals the identity there. For any tangent vector $v \in T_{X_0}\mathcal{M}^K$, differentiating this identity gives

$$\mathrm{d}g_0\big|_{F_K(X_0)}\Big(\mathrm{d}F_K\big|_{X_0} v\Big) = v. \tag{18}$$

Here d denotes the differential of the indicated map. If fusion sent a nonzero tangent vector to zero, the left side would vanish while the right side would not. Its tangent restriction must therefore preserve all Kd independent directions. Since its output lies in $\mathbb{R}^D$, the rank inequality is

$$Kd = \mathrm{rank}(J_{\mathcal{M}} m_0) \le \mathrm{rank}(J_{\mathcal{M}} F_K) \le D. \tag{19}$$

Here $J_{\mathcal{M}}$ denotes restriction to $T_{X_0}\mathcal{M}^K$. Hence $D \geq Kd$ is a necessary capacity condition for smooth local recovery of the complete clean tuple. When $D < Kd$, at least one intrinsic direction must be lost. When $D \geq Kd$, preserving these directions becomes dimensionally possible. This distinction motivates the controlled toy experiment, which varies d, D, and K. Actual recovery additionally depends on the local separation developed by the fusion map. For MF-AE, apply the same argument to a smooth deterministic tuple-to-latent map $B_K : \mathcal{M}^K \to \mathbb{R}^{D_{\mathrm{AE}}}$. An ideal smooth decoder satisfying $\mathcal{D}_K \circ B_K = \mathrm{id}$ would require $D_{\mathrm{AE}} \geq Kd$. Here D_{AE} denotes the shared latent dimension. This applies the same capacity principle to joint reconstruction, consistent with the placement of fusion and unfusion around the autoencoder.

Stable local unfusion also calls for distinguishability of nearby clean tuples, for example

$$\|F_K(X) - F_K(X')\| \geq c\|X - X'\|, \qquad c > 0, \tag{20}$$

within a local neighborhood of $\mathcal{M}^K$. The bound gives an inverse sensitivity of at most $1/c$ on the local image, so that separated clean tuples remain separated in fused space. To see why full tangent rank yields a local inverse, choose local coordinates $u \in \mathbb{R}^{Kd}$ on $\mathcal{M}^K$. A rank-Kd derivative contains a nonsingular $Kd \times Kd$ minor. Selecting those output coordinates gives a map with an invertible derivative, so the inverse function theorem supplies a local inverse for the selected map and hence for F_K on its image. Shrinking to a neighborhood with a bounded inverse derivative gives a finite local inverse Lipschitz constant, equivalent to a lower bound of the form above. The constant is local and can deteriorate as the derivative approaches rank deficiency. Slot-specific mappings $F_K(X) = \sum_{i=1}^{K} \psi_{K,i}(x^i)$ allow each sample to occupy different representation directions. Prediction training favors arrangements from which the slot-wise targets can be recovered. The capacity condition and this local separation condition play complementary roles. The toy, response, and weight-orthogonality experiments examine their learned counterparts.

Noise, target compatibility, and TFS. At noisy timesteps, consider $X_t = a_t X_0 + \sigma_t \epsilon$ with $\sigma_t > 0$ and $\epsilon \sim \mathcal{N}(0, I_{KD})$ independent of X_0. Conditional on any clean tuple, the Gaussian perturbation has positive density throughout $\mathbb{R}^{KD}$, so its mixture over clean tuples also has full support there. Hence the noisy input distribution occupies KD ambient dimensions even though the clean tuple lies on the Kd-dimensional manifold $\mathcal{M}^K$. For $K > 1$, the continuous map $F_K : \mathbb{R}^{KD} \to \mathbb{R}^D$ cannot be globally injective. To see this, an injective map could be composed with the coordinate inclusion of $\mathbb{R}^D$ into $\mathbb{R}^{KD}$. Invariance of domain would require the resulting image to be open in $\mathbb{R}^{KD}$, contradicting its containment in a lower-dimensional coordinate subspace. The relevant objects are its fibers

$$F_K^{-1}(z) = \{X \in \mathbb{R}^{KD} : F_K(X) = z\}. \tag{21}$$

Low fusion ambiguity requires compatible optimal targets within these fibers. When velocity is the target, this recovers the original fiber-wise vector-field consistency interpretation, in which the conditional mean slot-wise velocity should have small variation within a fiber. The relevant variation is measured across tuples identified by fusion, which may be distant in input space. This motivates comparing slot-wise fields for near-fiber pairs against pairs separated in fused space. Together with alignment to a single-sample reference, these comparisons examine whether sharing a representation is accompanied by compatible prediction behavior.

Noise and SNR affect $\Delta_t(K)$ through both the conditional distribution and the target parameterization. For example, under $X_t = a_t X_0 + \sigma_t \epsilon$ with $\sigma_t > 0$, rearranging the noising equation gives $\epsilon = (X_t - a_t X_0)/\sigma_t$. Conditioning on X_t leaves X_t, a_t, and σ_t fixed, so linearity of conditional expectation gives

$$m_t^{\epsilon}(X_t) = \frac{X_t - a_t \mathbb{E}[X_0 \mid X_t, t]}{\sigma_t}. \tag{22}$$

In the pure-noise limit $a_t = 0$, $\sigma_t = 1$, this target equals X_t. This example clarifies why scheduling depends on the prediction target as well as SNR. Even in a high-noise state, direct noise prediction can require detailed state information. In comparison, a clean-state target depends on the posterior mean of X_0. TFS must therefore balance the information demanded by the chosen target with the effect of prediction error on the subsequent update.

TFS uses this distinction to allocate shared computation along the trajectory. For a velocity predictor, compare $\widehat{X}_{t'} = X_t + h_t \widehat{v}_t(X_t)$ and $X_{t'}^* = X_t + h_t v_t^*(X_t)$ from the same state, with signed interval $h_t = t' - t$. Subtraction cancels X_t, and homogeneity of the norm gives

$$\|\widehat{X}_{t'} - X_{t'}^*\| = |h_t| \, \|\widehat{v}_t(X_t) - v_t^*(X_t)\|, \tag{23}$$

where $X^*_{t'}$ is the Euler update using the reference velocity v^*_t from the same input state. The step size directly scales the contribution of velocity error to the next state. This supplies the rationale for assigning larger fusion factors to smaller Flow Matching intervals. For DDPM/DDIM, coarse-to-fine generation motivates retaining more per-sample capacity near final refinement. The trajectory-dependent alignment in Fig. 3 and the schedule ablations evaluate this allocation principle in the learned models. When comparing prediction risks across fusion factors, per-slot normalization such as $\Delta_t(K)/K$ accounts for the changing tuple size.

Conditional grouping and slot separation. Fiber consistency compares different complete tuples mapped to a shared representation. Conditional grouping determines how the individual slots within each tuple can be separated by unfusion. Diverse class conditions supply additional cues for assigning predicted features to the appropriate slot. The benefit of mixed-condition grouping therefore complements compatibility across tuples. One concerns slot separation, while the other concerns shared prediction behavior. Same-condition samples retain distinct noisy states and targets, providing alternative cues for unfusion. Our class-conditional grouping experiment examines how the availability of these cues affects generation quality.

Scope of the analysis. The identities above describe fixed-timestep population risk under squared loss, and the clean-capacity argument assumes smooth maps on the clean manifold. The empirical analyses use learned reference fields and finite perturbations to examine the associated prediction geometry. End-to-end generation quality and the cumulative effects of fusion are assessed by the sampling metrics and TFS ablations.

D Empirical Geometric Analysis

Toy experiment To empirically validate the proposed geometric analysis, we construct a controlled setting in which the intrinsic dimension d and ambient dimension D are explicitly separated. Latent samples $\hat{x} \in \mathbb{R}^d$ are drawn from a low-dimensional distribution and embedded into the ambient space $\mathbb{R}^D$ through a fixed column-orthogonal projection $P \in \mathbb{R}^{D\times d}$ satisfying $P^\top P = I$, yielding observations $x = P\hat{x}$. The projection is unknown to the model, so diffusion models are trained entirely in the ambient space using lightweight MLP denoisers. To inspect the intrinsic structure, generated samples are projected back to the latent coordinates through $P^\top$. Within this setting, we train denoisers both with and without fusion and analyze the resulting fused dynamics under different fusion factors.

To examine clean-manifold capacity under fusion, we vary the relationship between the ambient dimension D and the fused intrinsic dimension Kd. Consistent with the necessary capacity condition derived in Sec. 2.3, recovery fails when the ambient space cannot represent the fused product manifold, while stable separation is observed for $D \geq Kd$ in the evaluated configurations (Fig. 4). This transition empirically supports the role of representation capacity in clean-sample recovery.

To examine representation alignment, we analyze the geometry induced by the fusion map in the ambient denoising space. We visualize fused representations after projection to the intrinsic coordinates through $P^\top$. Although the fused geometry does not initially coincide with the single-sample manifold, an affine alignment in the ambient space substantially reduces the discrepancy. In recoverable regimes, the aligned fused geometry closely matches the single-sample manifold structure, supporting preservation of the reference structure in the projected features.

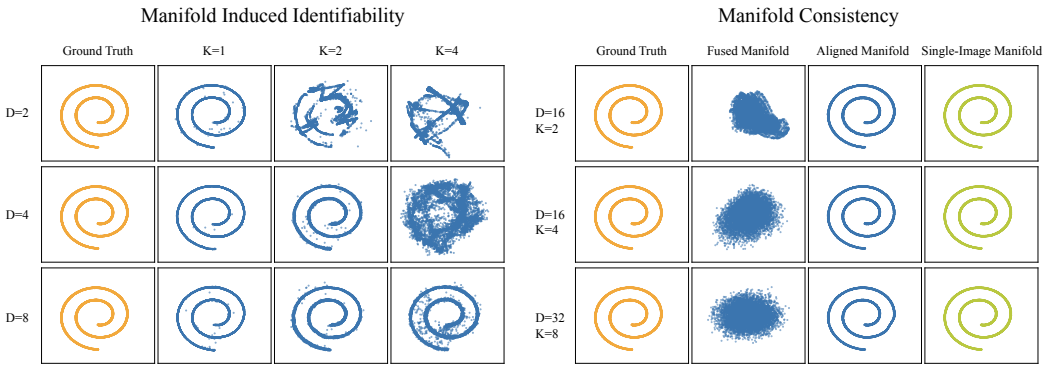


Figure 4: Clean-capacity and representation-alignment diagnostics under multi-sample fusion. *Left.* Swiss-roll samples as ambient dimension D and fusion factor K vary. *Right.* An affine transformation aligns the visualized fused representations with the single-image reference in this toy setting. The RMSEs are 4.83×10^{-4} for $(D{=}16, K{=}2)$, 7.03×10^{-4} for $(D{=}16, K{=}4)$, and 7.8×10^{-5} for $(D{=}32, K{=}8)$.

Local Smoothness and Empirical Lipschitz Analysis We further examine the local smoothness of the denoising dynamics through empirical local response ratios. For a denoiser f_t, we measure

$$\frac{\|f_t(x+\delta) - f_t(x)\|_2}{\|\delta\|_2}, \tag{24}$$

where δ denotes a small local perturbation around the current noisy sample. This ratio measures how sensitively the denoising output changes in the sampled perturbation direction.

We observe that the empirical ratios for $K{=}2$ and $K{=}4$ remain clearly above zero while staying well controlled, indicating that the denoiser remains responsive to the sampled input variations (Fig. 5). For $K{=}8$, the ratio is still positive, but noticeably smaller than those of $K{=}2$ and $K{=}4$. This suggests that local prediction differences become more attenuated under aggressive fusion. These results support the TFS design, which reduces fusion during sensitive refinement stages.

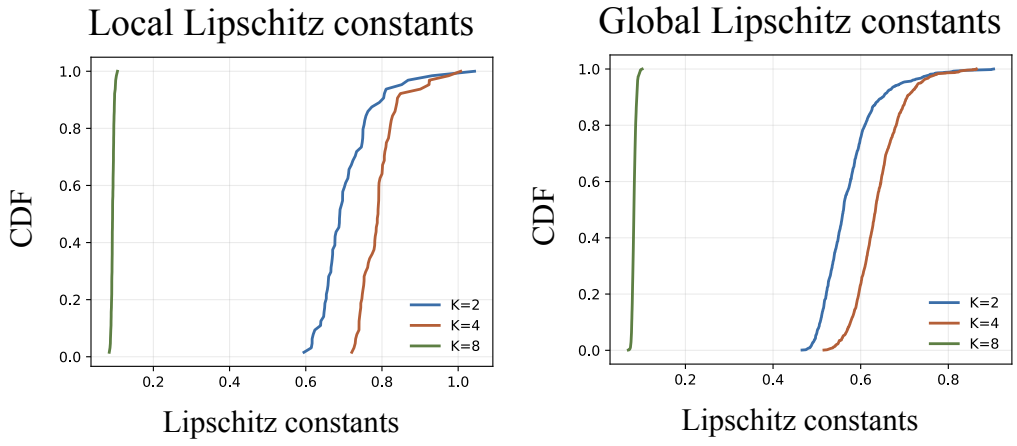


Figure 5: Empirical local response ratios under multi-sample fusion. We measure local Lipschitz ratios of the denoising dynamics under both near-neighbor and fusion-induced perturbations across fusion factors. The estimated ratios for $K{=}2$ and $K{=}4$ remain clearly positive while staying well controlled, showing nonzero output responses in the sampled directions. For $K{=}8$, the ratio remains positive but becomes noticeably smaller, indicating attenuated responses under more aggressive fusion in this diagnostic.

Interpolation Behavior under ManiFusion We analyze linear interpolation in the noise space. We randomly sample two CelebA images x_0^a, x_0^b and map them to the terminal diffusion time $t{=}T$ via the forward process, yielding x_T^a and x_T^b. We then construct interpolated noises

$$\tilde{x}_T(\lambda) = (1-\lambda)x_T^a + \lambda x_T^b, \quad \lambda \in [0,1], \tag{25}$$

and run the reverse process starting from $\tilde{x}_T(\lambda)$. We compare a standard baseline and ManiFusion ($K{=}2$). We visualize the resulting samples $\hat{x}_0(\lambda)$ along a grid of λ values. In Fig. 6, baseline often exhibits mode switching and non-smooth transitions. In contrast, the ManiFusion produces noticeably smoother morphing between the two identities, showing smoother interpolation behavior. These examples support coherent changes in generated appearance as the input noise varies.

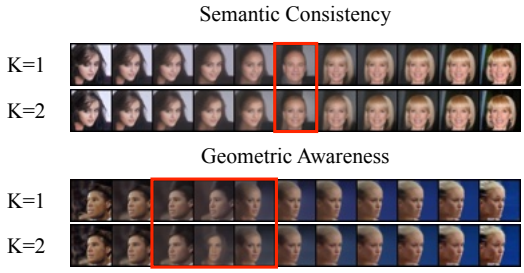


Figure 6: Interpolation behavior under ManiFusion on CelebA. The top panel shows semantic interpolation between two identities by interpolating noise and running the reverse process. The bottom panel shows geometric interpolation between pose/appearance factors. Compared with the baseline (K = 1), ManiFusion (K = 2) yields smoother transitions and fewer abrupt mode switches. Red boxes highlight representative transition regions.

Manifold Coverage under ManiFusion While ManiFusion slightly reduces precision, recall remains comparable to the baseline model. This suggests that the generative model continues to cover similar regions of the evaluation feature space despite signal fusion. To examine this hypothesis, we visualize real and generated samples in feature space using t-SNE. The visualization shows that samples from ManiFusion occupy a similar region of the displayed feature space as those from the baseline model, indicating comparable coverage(Fig. 7). Together, recall and these visualizations support comparable sample diversity in the evaluated setting.

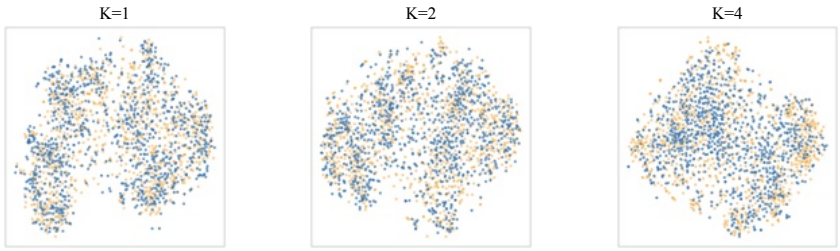


Figure 7: t-SNE visualization of feature distributions on CIFAR-10. The plots visualize the feature embeddings of real data (orange) and generated samples (blue) across different fusion factors. The panels show standard diffusion ($K = 1$) and ManiFusion with $K = 2$ and $K = 4$ from left to right.

E Extensions of ManiFusion

Large-Fusion Extension and Trajectory-Fusion Scheduling We extend the fusion factor to larger values and study TFS across all experiments in Sec 2. Specifically, we consider fusion factors up to $K = 8$ and evaluate schedules that combine $\{8, 4, 2, 1\}$ with different ratios across denoising timesteps. As shown in Table 5, Table 6 and Table 7, naive fusion with large K significantly degrades generation quality despite providing large throughput gains. *However, for MF-DiT, compared with increasing patch size for efficiency as in DiT-S/4, ManiFusion (-TFS, K=4) achieves similar throughput gains while preserving substantially better sample quality.* Moreover, ManiFusion reduces the fusion factor over the denoising process and enables the use of larger fusion factors without severe quality degradation. Notably, *using $K = 8$ with a uniform schedule* $(25\%, 25\%, 25\%, 25\%)$ *achieves both higher throughput and better generation quality compared to schedules that rely only on* $\{4, 2, 1\}$. This result highlights that large fusion factors early in denoising can further improve the efficiency–fidelity trade-off.

Table 5: Extended comparison of Vanilla, Ours (-TFS), and **Ours** across DDPM, DDIM, FM, and DiT on CIFAR-10 and CelebA. Ours (-TFS) uses a fixed fusion factor K, while **Ours** dynamically allocates fusion factors across denoising timesteps. For **Ours**, each tuple specifies the timestep allocation ratio for $K = \{8, 4, 2, 1\}$. Larger fusion factors and additional TFS schedules are evaluated only for DDPM. Best results are in **bold**, and second-best results are underlined.

CIFAR-10

Method	Variant	Throughput↑	MAC↓	Parameter	FID↓	sFID↓	IS↑	Precision↑	Recall↑
DDPM	Vanilla	1.71	6.07T	35.7M	**3.41**	4.40	9.20	**0.68**	0.58
	Ours (-TFS, $K = 2$)	3.17 (×1.85)	3.07T (50.6%)	+63.1K (0.18%)	5.93	4.58	8.67	0.63	0.58
	Ours (-TFS, $K = 4$)	6.31 (×3.69)	1.55T (25.5%)	+102.0K (0.29%)	18.27	6.54	7.41	0.56	0.55
	Ours (-TFS, $K = 8$)	**10.36** (×6.06)	**0.78T** (12.9%)	+179.8K (0.50%)	43.53	13.95	5.47	0.62	0.45
	Ours (0%, 50%, 25%, 25%)	3.31 (×1.94)	3.06T (50.4%)	+102.0K (0.29%)	3.88	**4.27**	**9.29**	0.66	**0.60**
	Ours (25%, 25%, 25%, 25%)	3.82 (×2.23)	2.87T (47.2%)	+179.8K (0.50%)	3.87	4.28	**9.29**	0.66	**0.60**
	Ours (50%, 25%, 12.5%, 12.5%)	5.00 (×2.92)	1.92T (31.7%)	+179.8K (0.50%)	8.53	5.25	8.74	0.61	**0.60**
DDIM (T=50)	Vanilla	35.47	303.44G	35.7M	**7.99**	6.66	**8.71**	**0.64**	0.55
	Ours (-TFS, $K = 2$)	65.39 (×1.84)	153.44G (50.6%)	+63.1K (0.18%)	13.34	7.16	8.11	0.59	0.56
	Ours (-TFS, $K = 4$)	**120.95** (×3.41)	**77.31G** (25.5%)	+102.0K (0.29%)	26.92	7.95	7.16	0.53	0.54
	Ours (0%, 50%, 25%, 25%)	66.80 (×1.88)	152.87G (50.4%)	+102.0K (0.29%)	10.35	**6.29**	8.58	0.60	**0.57**
FM	Vanilla	9.60	124.55G	55.7M	**2.74**	**3.84**	**9.47**	**0.65**	**0.63**
	Ours (-TFS, $K = 2$)	18.22 (×1.90)	62.89G (50.5%)	+185.9K (0.33%)	6.72	4.50	8.76	0.62	0.60
	Ours (-TFS, $K = 4$)	**32.98** (×3.44)	**31.69G** (25.4%)	+323.4K (0.58%)	21.11	7.10	7.35	0.58	0.53
	Ours (0%, 50%, 25%, 25%)	18.29 (×1.91)	62.70G (50.3%)	+323.4K (0.58%)	4.10	3.95	8.72	0.61	0.59
DiT	Vanilla	8.25	217.00G	32.48M	**10.58**	7.37	**10.17**	**0.75**	**0.40**
	Ours (-TFS, $K = 2$)	15.86 (×1.92)	114.25G (52.6%)	+1.04M (3.02%)	11.13	**6.45**	9.50	0.72	**0.40**
	Ours (-TFS, $K = 4$)	**29.88** (×3.62)	**60.00G** (27.6%)	+1.63M (5.02%)	18.59	8.68	8.74	0.68	0.34
	Ours (0%, 50%, 25%, 25%)	15.95 (×1.93)	113.84G (52.5%)	+1.63M (5.02%)	10.97	7.71	9.60	0.73	0.37

CelebA

Method	Variant	Throughput↑	MAC↓	Parameter	FID↓	sFID↓	IS↑	Precision↑	Recall↑
DDPM	Vanilla	0.71	15.56T	113.7M	**3.51**	6.27	2.77	**0.76**	0.50
	Ours (-TFS, $K = 2$)	1.28 (×1.80)	7.92T (50.9%)	+63.1K (0.06%)	4.73	6.98	2.70	0.75	0.46
	Ours (-TFS, $K = 4$)	2.34 (×3.30)	4.01T (25.8%)	+102.0K (0.09%)	9.15	10.34	2.64	0.70	0.40
	Ours (-TFS, $K = 8$)	**3.84** (×5.41)	**2.04T** (13.1%)	+179.8K (0.16%)	34.60	30.21	2.26	0.56	0.11
	Ours (0%, 50%, 25%, 25%)	1.33 (×1.87)	7.87T (50.6%)	+102.0K (0.09%)	3.81	**6.00**	2.77	0.74	0.51
	Ours (25%, 25%, 25%, 25%)	1.40 (×1.97)	7.38T (47.4%)	+179.8K (0.16%)	3.97	7.02	**2.83**	0.73	**0.52**
	Ours (50%, 25%, 12.5%, 12.5%)	1.98 (×2.79)	4.96T (31.9%)	+179.8K (0.16%)	7.72	8.51	2.76	0.67	0.49
DDIM (T=50)	Vanilla	14.56	778.00G	113.7M	**5.58**	8.37	2.81	**0.74**	0.46
	Ours (-TFS, $K = 2$)	26.05 (×1.79)	395.88G (50.1%)	+63.1K (0.06%)	7.43	9.07	2.75	0.71	0.42
	Ours (-TFS, $K = 4$)	**47.67** (×3.27)	**200.32G** (25.8%)	+102.0K (0.09%)	12.62	12.26	2.67	0.65	0.36
	Ours (0%, 50%, 25%, 25%)	26.83 (×1.84)	393.63G (50.6%)	+102.0K (0.09%)	6.52	**7.82**	**2.84**	0.68	**0.48**
FM	Vanilla	2.12	589.55G	169.1M	**2.80**	5.35	**3.23**	0.67	**0.60**
	Ours (-TFS, $K = 2$)	4.06 (×1.92)	297.24G (50.4%)	+185.9K (0.11%)	3.67	5.67	3.17	**0.68**	0.56
	Ours (-TFS, $K = 4$)	**7.65** (×3.61)	**149.59G** (25.4%)	+323.4K (0.19%)	7.15	7.24	3.00	0.62	0.52
	Ours (0%, 50%, 25%, 25%)	4.08 (×1.92)	296.49G (50.3%)	+323.4K (0.19%)	2.98	**5.20**	3.22	0.66	0.59

Combination with DeepCache Table 6 reports the combined throughput and quality of fusion and DeepCache, with and without TFS.

Table 6: Combination of ManiFusion (MF-UNet) w/w.o TFS with DeepCache on CIFAR-10 and CelebA. MAC reduction ratios and throughput speedups are shown in parentheses. *DDIM steps are reduced to match the same throughput setting.

CIFAR-10								
Setting	Variant	Throughput↑	MAC↓ (G)	FID↓	sFID↓	IS↑	Prec↑	Rec↑
DDIM	Vanilla	17.70	606.88	6.30	5.65	8.88	0.65	0.56
	+DeepCache	28.05 (×1.59)	390.18 (64.3%)	6.90	6.21	8.89	0.64	0.56
	+Step-Reduction	49.02* (×2.77)	207.88 (34.3%)	10.12	7.40	8.56	0.63	0.53
Ours (-TFS, $K = 2$)	Vanilla	33.48 (×1.89)	306.88 (50.6%)	11.21	6.12	8.30	0.59	0.57
	+DeepCache	51.19 (×2.89)	198.48 (32.7%)	11.24	6.48	8.29	0.59	0.56
Ours (-TFS, $K = 4$)	Vanilla	56.80 (×3.21)	154.63 (25.5%)	23.72	7.22	7.31	0.54	0.55
	+DeepCache	91.88 (×5.19)	100.43 (16.6%)	23.87	8.11	7.37	0.52	0.54
Ours	Vanilla	32.90 (×1.86)	307.76 (50.7%)	8.65	5.46	8.68	0.61	0.59
	+DeepCache	51.98* (×2.94)	197.38 (32.5%)	9.42	6.37	8.62	0.61	0.57
CelebA								
Setting	Variant	Throughput↑	MAC↓ (G)	FID↓	sFID↓	IS↑	Prec↑	Rec↑
DDIM	Vanilla	7.20	1556.01	4.17	6.70	2.82	0.74	0.50
	+DeepCache	9.81 (×1.36)	1075.81 (69.1%)	6.07	9.10	2.79	0.72	0.45
	+Step-Reduction	16.83* (×2.34)	572.58 (36.8%)	8.22	11.38	2.73	0.70	0.40
Ours (-TFS, $K = 2$)	Vanilla	13.25 (×1.84)	791.76 (50.9%)	6.17	7.71	2.76	0.71	0.45
	+DeepCache	16.96 (×2.36)	551.66 (35.5%)	8.43	10.50	2.71	0.69	0.40
Ours (-TFS, $K = 4$)	Vanilla	23.81 (×3.31)	400.63 (25.8%)	12.07	11.93	2.69	0.65	0.38
	+DeepCache	30.41 (×4.23)	280.63 (18.0%)	14.29	14.03	2.62	0.63	0.32
Ours	Vanilla	13.41 (×1.86)	787.26 (50.6%)	5.54	6.91	2.85	0.69	0.51
	+DeepCache	17.65* (×2.45)	547.18 (35.2%)	7.07	8.69	2.79	0.68	0.46

Scaling Fusion on ImageNet Table 7 compares fixed fusion factors and TFS on ImageNet using DiT-S/2, alongside the DiT-S/4 baseline.

Table 7: Extension of Ours(-TFS) to larger fusion factors and **Ours** on ImageNet using a DiT-S/2 backbone. Numbers in parentheses indicate throughput speedups and MAC reduction ratios relative to the baseline. For **Ours**, each tuple (e.g., $(0\%, 50\%, 25\%, 25\%)$) specifies the fraction of denoising timesteps allocated to fusion factors $K = \{8, 4, 2, 1\}$.

ImageNet								
Method	Throughput↑	MAC↓ (G)	Parameter	FID↓	sFID↓	IS↑	Prec↑	Rec↑
DiT-S/2 (baseline)	3.04	1067.50	32.9M	71.85	12.29	18.65	0.35	0.55
Ours (-TFS,K=2)	5.95 (×1.96)	562.50 (52.69%)	+1.33M (4.04%)	78.22	15.38	16.83	0.34	0.53
Ours (-TFS,K=4)	11.61 (×3.82)	286.88 (26.87%)	+1.94M (5.90%)	93.64	20.59	14.30	0.26	0.33
Ours (-TFS,K=8)	21.42 (×7.05)	153.75 (14.40%)	+2.56M (7.78%)	136.06	46.86	8.14	0.16	0.13
Ours (0%, 50%, 25%, 25%)	5.98 (×1.97)	551.95 (51.70%)	+1.94M (5.90%)	74.19	14.85	19.08	0.34	0.49
Ours (25%, 25%, 25%, 25%)	6.38 (×2.10)	520.03 (48.71%)	+2.56M (7.78%)	79.16	15.62	16.67	0.34	0.53
Ours (50%, 25%, 12.5%, 12.5%)	9.39 (×3.09)	351.29 (32.90%)	+2.56M (7.78%)	97.98	23.49	12.91	0.28	0.40

Extension to Latent Diffusion Models. We examine the applicability of ManiFusion to latent diffusion models by studying how fusion can be integrated into the denoising backbone. In particular, we evaluate MF-UNet, which applies fusion inside the UNet backbone, and compare it with the MF-AE design used in the main results. As shown in Table 8, ManiFusion (-TFS) provides significant computational savings and throughput gains (up to $3.87\times$) but leads to noticeable degradation in generation quality as the fusion factor increases. Compared to the MF-AE design presented in the main results, MF-UNet consistently yields worse fidelity across metrics such as FID and sFID. Qualitative comparisons between MF-UNet and MF-AE are provided in Table 17.

Table 8: MF-UNet in Latent Diffusion Models on CelebA-HQ.

	CelebA-HQ							
Method	Throughput	MAC(T)	Parameter	FID↓	sFID↓	IS↑	Prec↑	Rec↑
LDM (baseline)	0.54	96.38	362.3M	5.10	10.13	3.28	0.72	0.49
MF-UNet (K=2)	1.07 (×1.98)	48.67 (50.5%)	+159.8K (0.04%)	11.60	11.87	3.08	0.62	0.42
MF-UNet (K=4)	2.09 (×3.87)	24.61 (25.5%)	+319.6K (0.09%)	16.86	18.59	3.04	0.56	0.39

F Analysis of Fusion Behavior

Conditional Grouping and Intra-Tuple Separability We analyze how conditional grouping affects the separability of fused samples in MF-DiT. During training, fused inputs can be constructed either by grouping samples that share the same condition or by randomly mixing conditions within a fusion group. Table 9 shows that random condition mixing consistently outperforms uni-class grouping. This observation supports the interpretation that conditional diversity provides clearer slot-specific cues and eases index-conditioned unfusion. This is an intra-tuple separation effect, whereas fiber consistency concerns compatibility between predictions for different complete tuples sharing a fused representation. Same-condition grouping remains effective in the evaluated setting because individual trajectories still have distinct noisy states and targets, although fewer condition-specific cues can make separation harder.

Table 9: Effect of conditional grouping in ManiFusion. We compare grouping samples from the same class (uni-class) versus mixing different classes within a fused group (multi-class). Numbers in parentheses indicate throughput speedups and MAC reduction ratios relative to the baseline.

Method	Variant	Throughput↑	MAC↓ (G)	FID↓	sFID↓	IS↑	Prec↑	Rec↑
Baseline	-	34.21	108.47	10.58	7.37	10.17	0.75	0.40
Ours (-TFS, K=2)	uni-class	64.87 (×1.90)	56.41 (52.0%)	14.89	8.49	8.87	0.69	0.37
Ours (-TFS, K=2)	multi-class	64.61 (×1.89)	56.41 (52.0%)	11.13	6.45	9.50	0.72	0.40
Ours (-TFS, K=4)	uni-class	122.69 (×3.58)	29.56 (27.3%)	23.54	9.65	8.39	0.61	0.33
Ours (-TFS, K=4)	multi-class	122.24 (×3.57)	29.56 (27.3%)	18.59	8.68	8.74	0.68	0.34

G Ablation Study

We conduct an ablation study on the kernel size used in the fusion stage. As shown in Table 10, increasing the kernel size consistently degrades generation quality across FID, sFID, and IS. We attribute this behavior to stronger spatial entanglement introduced by larger kernels, which mixes information from different inputs more aggressively during fusion and makes unfusion harder. Across metrics, compact kernels reduce entanglement and preserve sample quality under fusion.

Table 10: Ablation on fusion kernel size. Larger kernels increase spatial entanglement, degrading sample quality. Smaller kernels (e.g., size 3) yield the best performance across FID, sFID, and IS.

Kernel	FID↓	sFID↓	IS↑	Prec↑	Rec↑
3	**5.93**	**4.58**	**8.67**	**0.63**	0.58
5	6.94	4.74	8.58	0.61	**0.59**
7	7.47	4.74	8.33	0.62	0.58

H Comparison with Throughput-Oriented Baselines

Comparison with a Naive Patch-Based Baseline. We first compare ManiFusion with a naive patch-based strategy that attempts to increase throughput by spatially compositing multiple inputs. Using the same model architectures and dataset configurations described in Section B, we design a naive multi-sample strategy where each noisy input is first downscaled to one-quarter resolution, concatenated into a 2×2 grid, processed once by the diffusion model, and then split and upscaled back to full resolution. This simple approach, however, severely degrades generation quality. As illustrated in Table 11, the patch-based method introduces artifacts, unnatural boundaries, and facial/background blur, while ManiFusion preserves texture and color consistency.

Quantitative results in Table 12 further confirm this gap. When applied to a standard DDPM, the patch-based method suffers a dramatic quality drop—on CIFAR-10, FID deteriorates from 18.27 (ManiFusion) to 91.71, and IS falls from 7.41 to 4.55. On CelebA, ManiFusion achieves an FID of 9.50 versus 26.07 for the grid approach, and an IS of 2.58 compared to 2.37. These findings demonstrate that our fusion–unfusion modules not only preserve and disentangle multi-sample information more effectively than naive patch compositing but also scale reliably to higher throughput settings. Together, the qualitative and quantitative evidence underscores the scalability of ManiFusion and its potential to unlock further efficiency gains without compromising visual fidelity.

Table 11: Qualitative comparison of ManiFusion ($K = 4$) versus a naive 2×2 patch-based baseline on CIFAR-10 and CelebA. ManiFusion maintains sharper details and avoids mosaic artifacts.

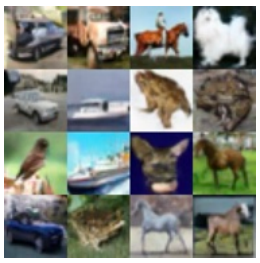
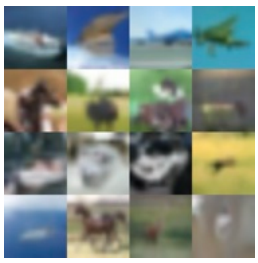
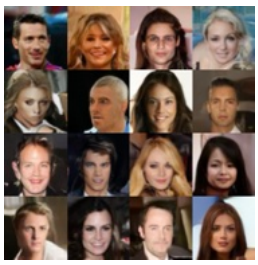
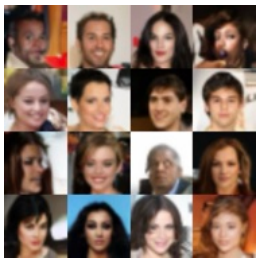
CIFAR-10		**CelebA**	
ManiFusion (K=4)	Patch-Based	ManiFusion (K=4)	Patch-Based
			

Table 12: Quantitative performance of ManiFusion ($K = 4$) compared with a naive patch-based baseline on CIFAR-10 and CelebA. Metrics include FID, sFID, IS, Precision, and Recall. ManiFusion significantly outperforms the naive approach.

	CIFAR-10					**CelebA**				
Method	FID↓	sFID↓	IS↑	Prec↑	Rec↑	FID↓	sFID↓	IS↑	Prec↑	Rec↑
ManiFusion (K=4)	18.27	6.54	7.41	0.56	0.55	9.50	10.66	2.58	0.70	0.38
Patch-Based	91.71	52.16	4.55	0.35	0.07	26.07	37.34	2.37	0.36	0.12

Comparison with Capacity-Reduced Models. Beyond comparisons to naive compositing, we also evaluate whether ManiFusion simply mimics the effect of reducing model size to improve throughput. To this end, we construct two reduced-computation U-Net variants trained on CIFAR-10 at $K = 2$. The first is a *channel-reduced* model, which decreases the number of feature maps per layer by modifying the channel multipliers. The second is a *depth-reduced* model, which shortens the network depth by removing residual blocks. These variants allow us to compare shared computation with reductions in backbone capacity at comparable computational budgets.

Detailed configurations are listed in Table 13, and quantitative results are reported in Table 14. The reference in this comparison is ManiFusion with the full backbone at $K = 2$. Both reduced models require more MACs and achieve lower throughput than this reference. FID increases from 5.93 for ManiFusion to 7.59 for the channel-reduced model and 6.55 for the depth-reduced model, with corresponding declines in IS and recall.

This comparison distinguishes two ways of reducing per-sample computation. Smaller backbones reduce the capacity used to process inputs, whereas ManiFusion shares the computation of the full backbone across multiple samples. In the tested configurations, the latter achieves better quality at lower MAC counts and higher throughput. These results support retaining backbone capacity through fusion as an effective alternative to reducing channels or depth.

Table 13: Backbone configurations for the CIFAR-10 DDPM comparison between ManiFusion and the channel-reduced and depth-reduced variants.

	CIFAR-10 (DDPM)		
	Full Backbone	Channel-Reduced	Depth-Reduced
Channels	128	128	128
Channel Multiplier	1,2,2,2	1,1,1,2	1,1,2
Number of Heads	1	1	1
Activation Function	SiLU	SiLU	SiLU
Diffusion Steps	1000	1000	1000
Noise Schedule	linear	linear	linear
Beta Start	0.0001	0.0001	0.0001
Beta End	0.02	0.02	0.02
Dropout	0.1	0.1	0.1
Learning Rate	2e-4	2e-4	2e-4
Batch Size	128	128	128

Table 14: Performance of ManiFusion ($K = 2$) and the capacity-reduced variants on CIFAR-10 with DDPM. ManiFusion achieves better FID, sFID, IS, precision, and recall at lower MAC counts and higher throughput in this comparison.

	CIFAR-10 (DDPM)						
Method	Throughput	MAC	FID↓	sFID↓	IS↑	Prec↑	Rec↑
ManiFusion ($K = 2$)	3.17	3.07T	5.93	4.58	8.67	0.63	0.58
Channel-Reduced	2.75	3.26T	7.59	4.78	8.25	0.62	0.55
Depth-Reduced	2.54	3.72T	6.55	4.65	8.47	0.63	0.57

I Qualitative Results

We organize the qualitative comparisons by evaluation setting. We first compare fusion factors across samplers and fusion designs for latent diffusion, then examine throughput scaling.

Samples across Samplers and Fusion Factors Tables 15 and 16 show random samples from DDPM, DDIM, and Flow Matching on CIFAR-10 and CelebA for $K = 1, 2, 4$ and TFS. These comparisons complement the quantitative evaluation of sample quality across samplers and fusion factors.

Table 15: Randomly generated samples from DDPM, DDIM (T=50), and Flow Matching on **CIFAR-10** for $K = 1, 2, 4$ and TFS. ManiFusion maintains coherence and diversity across different samplers.

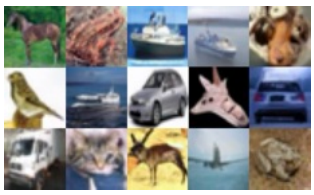
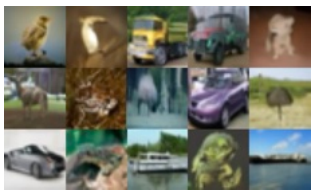
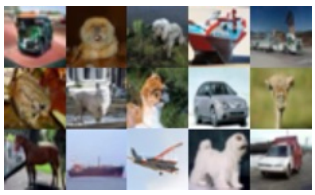
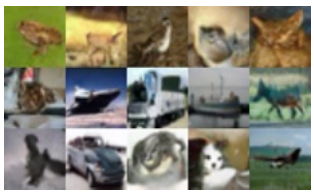
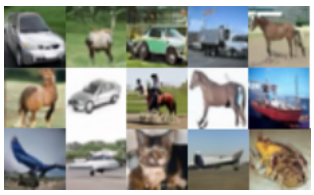
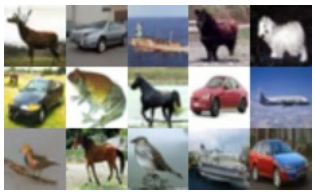
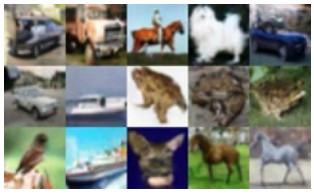
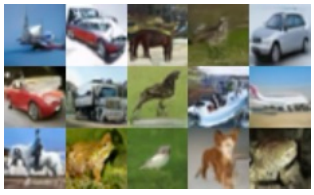
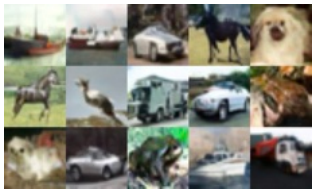
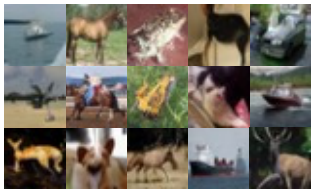
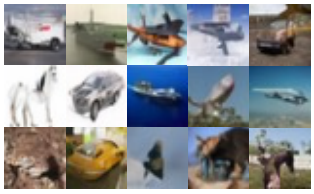
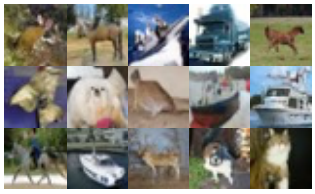
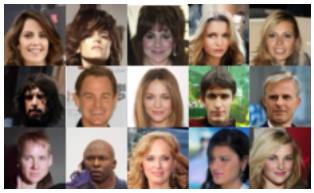
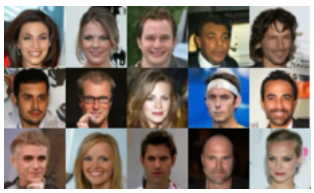
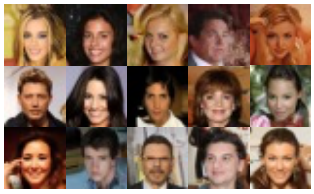
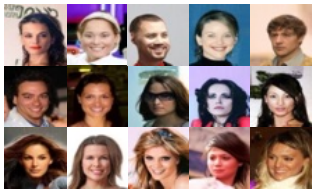
K	DDPM	DDIM (T=50)	FM
1			
2			
4			
TFS			

Table 16: Randomly generated samples from DDPM, DDIM (T=50), and Flow Matching on **CelebA** for $K = 1, 2, 4$ and TFS. Results confirm that ManiFusion preserves sample fidelity and diversity across datasets.

K	DDPM	DDIM (T=50)	FM
1			
2			
4			
TFS			

Fusion Designs for Latent Diffusion Table 17 compares the vanilla LDM with MF-AE and MF-UNet at $K = 2$ and $K = 4$, complementing the latent diffusion results above. For ImageNet, qualitative samples are omitted since the baseline DiT-S/2 configuration used in our experiments yields FID values above 50, rendering visual comparison uninformative.

Table 17: Qualitative comparison of vanilla LDM, MF-AE, and MF-UNet in latent diffusion models. Each fusion design is shown at $K = 2$ and $K = 4$.

LDM (Vanilla)	MF-AE		MF-UNet	
$K = 1$	$K = 2$	$K = 4$	$K = 2$	$K = 4$
				

Quality across Throughput Levels Tables 18 and 19 compare the baseline, fixed fusion ($K = 2$), and TFS at matched throughput levels on CIFAR-10 and CelebA. Columns follow the same order in both tables, from higher throughput to lower throughput.

Table 18: Qualitative scaling with throughput on **CIFAR-10**, showing random samples from ManiFusion (K=2) and TFS using DDIM with timesteps 10, 20, 50, and 100. Throughput ranges from 66.80–565.03 img/s.

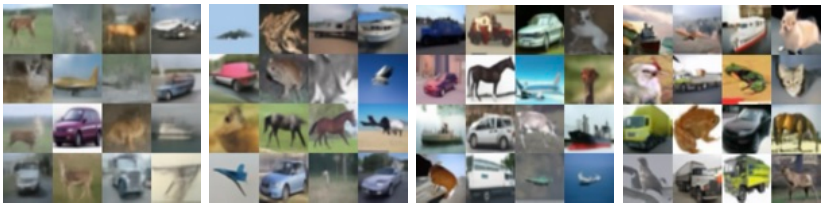
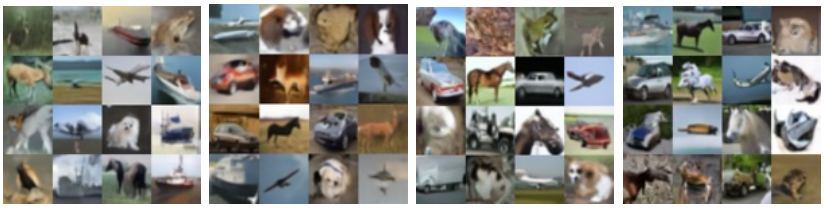
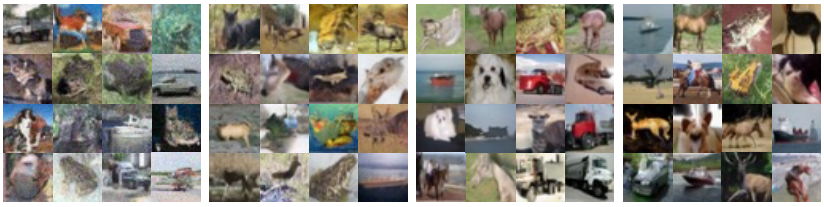
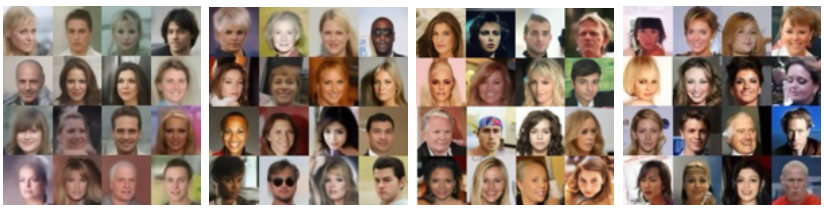
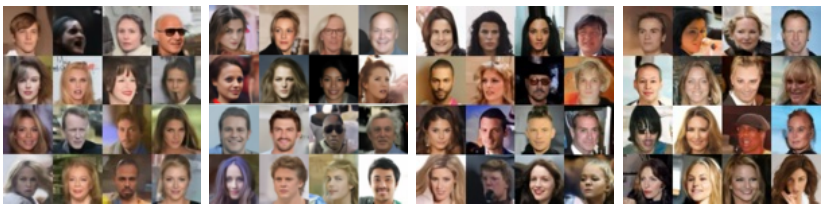
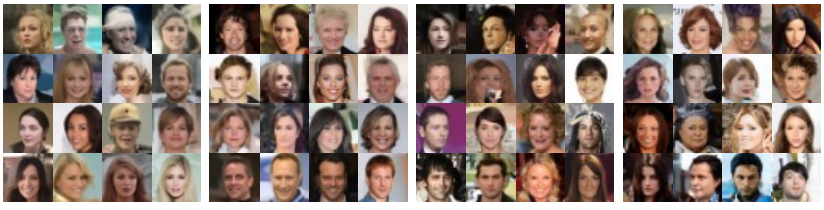
Throughput	565.03 img/s	317.89 img/s	156.56 img/s	66.80 img/s
Baseline				
ManiFusion (K=2)				
TFS				

Table 19: Qualitative scaling with throughput on **CelebA**, showing random samples from ManiFusion (K=2) and TFS using DDIM with timesteps 10, 20, 50, and 100. Throughput ranges from 26.83–228.77 img/s.

Throughput	228.77 img/s	127.92 img/s	66.90 img/s	26.83 img/s
Baseline				
ManiFusion (K=2)				
TFS				

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In both the abstract and the introduction, we underscored the limitations in prior works addressing the acceleration method of diffusion model inference. In response, we introduce an orthogonal approach to existing works.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work are discussed at the end of the conclusion (Section 6).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The assumptions underlying the geometric analysis are stated in Section 2.3, and the corresponding derivations and proofs are provided in Appendix C. The experiments provide empirical evidence for the behavior discussed in the analysis.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides experimental settings and results across various frameworks. Table 1,2,3 include key informations necessary for reproducing experiments suach as model structure, number of parameters, and performance metrics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide code in supplementary.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The "Implementation Details" section explains how the models were implemented and metrics for performance evaluation are clearly defined both in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We did not perform repeated experimental runs because of computational constraints, so we do not report error bars, standard deviations, or statistical significance tests. Generative-quality evaluation uses 50k generated images per setting, but this sample count does not establish variability across independent runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In appendix, we state that measurements are performed on a single NVIDIA A6000 GPU. Additionally, Table 1 provides detailed computational cost(MAC) and throughput for each model.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The ManiFusion approach proposed in the paper is a conceptual methodology to improve inference efficiency of diffusion models and does not contain content that violates NeurIPS ethic guidelines. No issues related to privacy, social bias, or potential malicious use are existing.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: The paper focuses on conceptual contributions and does not include a seperate discussion section on the positive/negative societal impacts of the proposed method.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper proposes a methodology to improve the efficiency of existing diffusion model frameworks and does not release new models or datasets with misuse risks. Therefore, no special safeguards are necessary.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets are pubically available.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper doesn't release new datasets, code or pre-trained models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research is entirely algorithmic and computational in focus and does not involve human participants or crowdsourcing experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve hunman subjects, so IRB approval or equivalent review is not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is only used for writing and editing. There are no components in the development of the methodology in this research that were obtained from an LLM.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.