FLOW ALONG THE K-AMPLITUDE FOR GENERATIVE MODELING

Anonymous authors
Paper under double-blind review

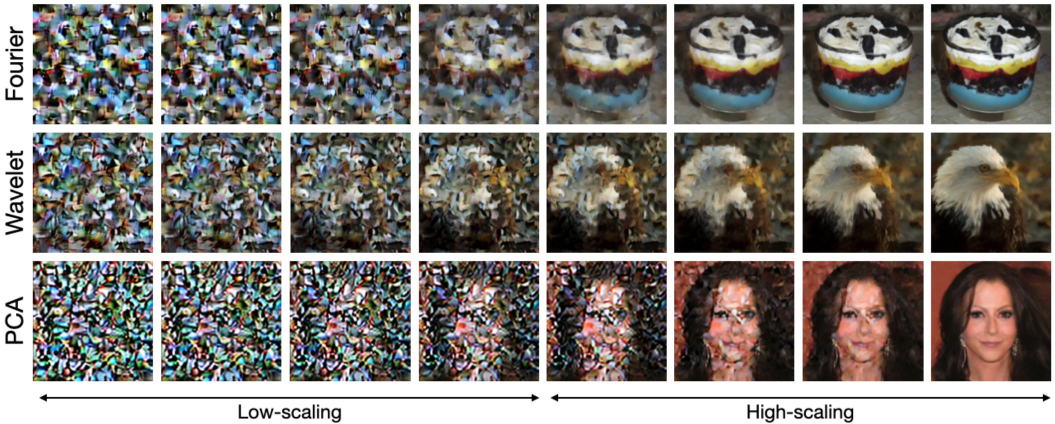


Figure 1: Unconditional generation using K-Flow using three types of K-amplitude decomposition.

ABSTRACT

In this work, we propose a novel generative learning paradigm, K-Flow, an algorithm that flows along the K-amplitude. In physics, k is a measure to organize the frequency bands of objects, and the amplitude is the norm of projected coefficients. By incorporating the K-amplitude decomposition, K-Flow enables flow matching across scaling as time. We discuss three venues of six properties of K-Flow, from theoretical foundations, energy and temporal dynamics, and practical applications, respectively. Specifically, from the practical usage perspective, K-Flow allows steerable generation by controlling the information at different scales. To demonstrate the effectiveness of K-Flow, we conduct experiments on unconditional image generation and class-conditional image generation. Additionally, we conduct three ablation studies to demonstrate how K-Flow steers scaling to effectively control the resolution of image generation.

1 INTRODUCTION

Generative Artificial Intelligence (GenAI) represents a pinnacle achievement in the recent wave of AI advancements. This field has evolved from foundational methods such as autoregressive models (AR) [48], energy-based models (EBMs) [7, 20, 23, 32, 54], variational auto-encoders (VAEs) [29], and generative adversarial networks (GANs) [18], to the most cutting-edge flow-matching (FM) framework [3, 36, 40].

Among these, flow matching (FM) stands out as a density transport method that converts an initial simple distribution into a complex target distribution through continuous-time flow dynamics. For instance, in the context of image generation, FM learns to map a random Gaussian distribution to the pixel-space distribution of images. This process, termed continuous *flow*, is governed by a localized *k-dependent vector field* (or velocity field) and produces a *time-dependent density path*, which represents the evolution of the probability distribution over time. As a versatile framework, FM can incorporate a diffusion density path, linking it to established methods such as denoising score matching (DSM)[53, 59] and the denoising diffusion probabilistic model (DDPM)[24].

Key Concepts. We first introduce several core concepts. The *scaling* and its parameter k can be interpreted as a measure to organize the frequency bands of physical objects or processes, and *amplitude* refers to the norm of coefficients obtained after projecting data concerning the scaling k, which we term the K-*amplitude space*, or equivalently, *scaling-amplitude space*. The underlying intuition behind the utility of K-amplitude space is that multi-scaling modeling inherently aligns more naturally with data structures in the K-amplitude space, *i.e.*, lower k tend to have higher amplitudes, as observed in multi-resolution image modeling [1].

Our Method. Such an understanding of scaling and K-amplitude space inspires a new paradigm for generative modeling, which we term **K Flow Matching (K-Flow)**. In essence, K-Flow performs flow along the K-amplitude. There are two main components in K-Flow, and the first is the K-amplitude decomposition. The K-amplitude decomposition encompasses a family of transformations, and in this work, we explore three types: Wavelet, Fourier, and principal component analysis (PCA) decomposition, as illustrated in Figure 1. Building on this, the second component in K-Flow is its flow process. K-Flow applies a K-amplitude transformation to project data from the spatial space into the K-amplitude space, learns a time-dependent velocity field in this space accordingly, and subsequently maps it back to the spatial space for velocity matching. A detailed pipeline is provided in Figure 2. Next, we will discuss the strengths of K-Flow through six properties, which can be organized into three categories: theoretical foundations (properties a & b), energy and temporal dynamics (properties c & d), and practical applications (properties e & f).

Properties of K-Flow. We introduce six main properties of K-Flow below, and please refer to Appendix C for more detailed discussions. In summary, K-Flow (a) provides a first-principle way to organize the scaling k, (b) enables multi-scale modeling in the K-amplitude space, (c) supports a well-defined scale along with energy, (d) interprets scaling as time, (e) supports the fusion of intra-scaling and inter-scaling modeling, and (f) supports explicit steerability.

Our Results. We conduct experiments on image generation to verify the effectiveness of K-Flow. Quantitatively, K-Flow achieves competitive performance in both unconditional and class-conditional image generation. Qualitatively, we conduct three ablation studies to demonstrate the steerability of K-Flow: controllable class-conditional generation and scaling-controllable generation.

2 BACKGROUND

2.1 SCALING PARAMETER k, AMPLITUDE, AND K-AMPLITUDE DECOMPOSITION

Our data generation framework leverages the implicit hierarchical structure of the data manifold. By ‘implicit’, we refer to the hierarchical characteristics that emerge when a generalized K-amplitude decomposition is applied, transitioning the representation from the original data space to the K-amplitude space. Illustrations are in Figure 2.

More formally, we represent data as a signal $\phi : \mathbb{R}^d \to \mathbb{R}^m$, or a finite discretization of $\mathbb{R}^d$ and $\mathbb{R}^m$, where this signal function is equivalent to a vector. For example, in the case of image data, each pixel can be viewed as a signal mapping from x-y-RGB coordinates to a pixel intensity value, *i.e.*, $\mathbb{R}^3 \to \mathbb{R}^1$. An alternative approach is to consider data as a high-dimensional vector $\mathbb{R}^{d \times m}$. However, treating data as signal functions provides a more natural fit in this work.

Without loss of generality, we take $m = 1$ for illustration. A K-amplitude decomposition involves the decomposition of a function using a complete basis set $\{\boldsymbol{e}_j\}_{j=1}^n$, where n can be infinite. We introduce a scaling parameter k, which partitions the set $\{\boldsymbol{e}_i\}_{i=1}^n$ into subsets: $\{\boldsymbol{e}_i\}_{i=1}^n = \bigcup_k \{e_k\}$, each with n_k basis. Hence, signal ϕ is expressed as:

$$\phi = \sum_k \phi_k, \tag{1}$$

where $\phi_k := \sum_{j=1}^{n_k} (\phi \cdot \boldsymbol{e}_{jk}) \boldsymbol{e}_{jk}$ for $\boldsymbol{e}_{jk} \in \{e_k\}$. Inspired by the concept of frequency amplitude, we also refer to the norm of ϕ_k as the K-amplitude. It is important to note that k is termed the scaling parameter because it implies that a well-structured decomposition should ensure that the amplitude decays with increasing k [17].

We define K-amplitude decomposition (or equivalently, K-amplitude transform) $\mathcal{F}$ as the map that sends ϕ to the collection of ϕ_k, and denote the collection of all $\{(\phi \cdot \boldsymbol{e}_{jk}) \boldsymbol{e}_{jk}\}_j$ as $\mathcal{F}\{\phi\}(k)$. Then, $\mathcal{F}\{\phi\} := \bigcup_k \mathcal{F}\{\phi\}(k)$. We further assume that this transform has an inverse, denoted by $\mathcal{F}^{-1}$.

2.2 EXAMPLE: FOURIER AMPLITUDE DECOMPOSITION

Suppose the data $\phi : \mathbb{R}^3 \to \mathbb{R}$, is drawn from a certain function distribution p_{data}. The challenge of directly fitting the distribution p_{data} is often complex and computationally demanding. Fourier frequency decomposition, however, offers a powerful technique to address this challenge by transforming ϕ into the Fourier space or Fourier domain. In what follows, we will use the terms 'space' and 'domain' interchangeably.

By applying Fourier frequency decomposition, we express ϕ as a sum of its frequency components. This transformation can potentially unveil the hidden structure within the distribution p_{data}, which is not apparent in the spatial or time domain, and it is thus beneficial for understanding the underlying patterns in the data manifold. To illustrate, the continuous Fourier transform $\mathcal{F}$ of data $\phi(x, y, z) : \mathbb{R}^3 \to \mathbb{R}$ is expressed as: $\mathcal{F}\{\phi\}(k_x, k_y, k_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \phi(x, y, z)\, e^{-2\pi i(k_x x + k_y y + k_z z)}\, dx\, dy\, dz$. After this transformation, the spatial variables (x, y, z) are converted into frequency variables (k_x, k_y, k_z), thereby representing the data in the frequency domain.

Note that the Fourier frequency is characterized by the high-dimensional vector representation (k_x, k_y, k_z). For our purposes, our aim is to distill the notion of frequency into a one-dimensional scaling parameter. Namely, we define the scaling parameter k as the diameter of the expanding ball in Fourier space: $k = \sqrt{k_x^2 + k_y^2 + k_z^2}$. This definition of k provides a simple index that captures the overall scaling of the frequency components in all directions. Moreover, we can decompose the Fourier transform $\mathcal{F}\{\phi\}$ into groups indexed by the scaling index k:

$$\mathcal{F}\{\phi\}(k) = \bigcup_{\sqrt{k_x^2+k_y^2+k_z^2}=k} \mathcal{F}\{\phi\}(k_x, k_y, k_z). \tag{2}$$

Intuitively, $\mathcal{F}\{\phi\}(k)$ represents the set of all frequency components that share the same scaling k. This grouping allows us to examine the contributions of various spatial frequencies of ϕ when viewed through the lens of frequency k. Furthermore, ϕ_k is just the summation of $\mathcal{F}\{\phi\}(k)$.

On the other hand, we can recover ϕ from $\mathcal{F}\{\phi\}$, because the Fourier transform is an invertible operation: $\phi = \mathcal{F}^{-1}\mathcal{F}\{\phi\}$. Such an invertibility establishes the Fourier transform as a valid example of K-amplitude decomposition. For discrete data, which inherently possess a highest resolution, the variables (k_x, k_y, k_z) are situated on a discrete lattice rather than spanning the entire continuous space. Consequently, the scaling parameter k is itself discrete and bounded.

2.3 FLOW MATCHING

In this work, we primarily focus on the flow matching (FM) generative models and their families [3, 36, 40]. In FM, the flow Ψ_t is defined by solutions of an ordinary differential equation (ODE) system with a time-dependent vector field $\boldsymbol{v}$:

$$\frac{d}{dt}\Psi_t(x) = \boldsymbol{v}_t(\Psi_t(x)), \tag{3}$$

and we focus on the probability transport aspects of Ψ_t. In particular,, the flow provides a means of interpolating between probability densities within the sample space. Suppose Ψ_t follows an initial probability p_0, then for $t > 0$, Ψ_t induces a probability measure p_t: $p_t(B) = p_0(\Psi_t^{-1}(B))$, where B is a measurable set. Assume that Ψ_t is differentiable, and define a surrogate velocity at time t as $v_t(x_t, \theta)$ using a deep neural network with parameter θ. Then the vector field matching loss is defined as:

$$\mathcal{L}_{\text{FM}} := \int\int_0^1 dx_0\, dt \left\| \frac{d\Psi_t}{dt}(x_t) - v_t(x_t, \theta) \right\|^2. \tag{4}$$

By aligning the learned vector field with the true gradient field of the frequency decomposition, this loss function ensures robust approximation and reconstruction of the data. Additionally, every interpolation $\pi(x_0, x_1)$ with a time-continuous interpolating function $f_t(x_0, x_1)$ between probabilities p_0 and p_1 induces a vector field v_t through the continuity equation:

$$\frac{\partial p_t(x_t)}{\partial t} = -\nabla_x\left(p_t(x_t)v_t(x_t)\right), \tag{5}$$

and v_t is explicitly expressed as: $v_t = \frac{1}{p_t}\mathbb{E}_{\pi(x_0,x_1)}[\frac{\partial f_t(x_0,x_1)}{\partial t}]$. Although explicit matching of v_t via the continuity equation is intractable, flow matching permits a conditional version:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\pi(x_0,x_1)}\int_0^t dt \left\| \frac{\partial f_t(x_0, x_1)}{\partial t} - v_t(x_t, \theta) \right\|^2 + \text{constant}. \tag{6}$$

As detailed in Section 3, our framework reinterprets the time variable t as scaling k.

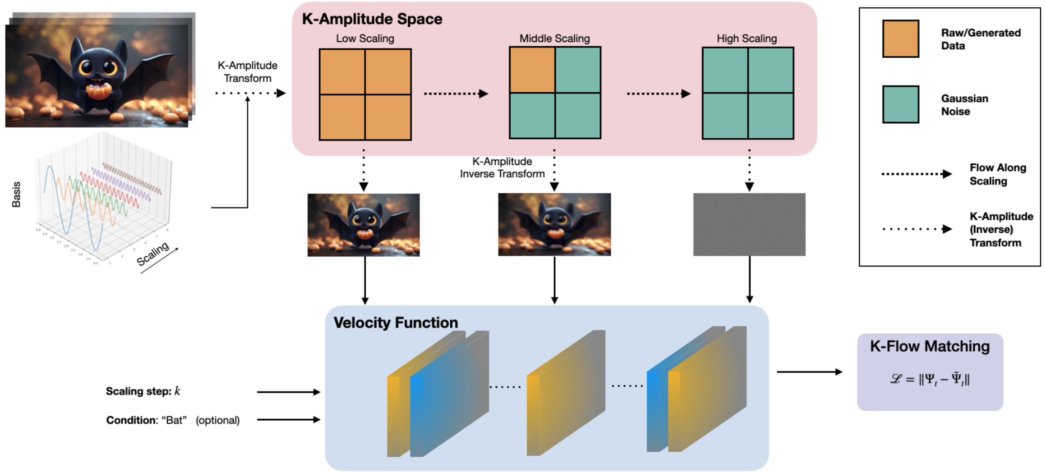


Figure 2: Pipeline of K-Flow.

3 METHODOLOGY: K-FLOW

3.1 K-AMPLITUDE INTERPOLANTS

According to the concept of stochastic interpolants [4], all flow models can be viewed as constructing stochastic paths that interpolate between a known tractable prior distribution and an unknown target distribution, including flow matching [36], rectified flow [40], and denoising diffusion [24]. By incorporating the scaling parameter k for K-amplitude decompositions, we can formulate a stochastic interpolant that gradually emerges each amplitude component from white noise. Given that k traverses monotonically from zero to a maximum value $k_{\max}$, this process draws a natural analogy to continuous normalizing flows. Since we require $\mathcal{F}$ to be invertible, we can reconstruct the data once the complete spectrum in the K-amplitude space is generated.

To build a **continuous flow** Ψ_k out of Equation (1), we explore two paradigms in designing the interpolants: (1) We generalize the original discrete-valued k to continuous values; (2) We ensure that the generation flow, which maps the white noise to the real data, remains invertible such that no information is lost throughout the process. Still taking the three-dimensional signal $\phi(x, y, z)$ and the Fourier transform $\mathcal{F}\{\phi\}$ as an example, we realize the second ingredient by introducing noise padding ϵ for each k and define the **discrete flow** φ_k as follows:

$$\varphi_k = \mathcal{F}^{-1}\Big(\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\leq k} \cdot \mathcal{F}\{\phi\}(k_x, k_y, k_z) + \big(1 - \mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\leq k}\big) \cdot \epsilon\Big), \tag{7}$$

where $\mathbb{I}$ is the indicator function that selects K-amplitude components up to the scaling step k. This formulation ensures that for each step k, the reconstruction incorporates the relevant K-Flow components of data ϕ and pads the rest with noise ϵ. Here, the noise ϵ is independently drawn from a known distribution (*e.g.*, uniform or Gaussian) for each coordinate (k_x, k_y, k_z). Through this construction, ϕ_k serves as a stochastic interpolant for the data ϕ, ensuring that: $\lim_{k\to k_{\max}} \varphi_k = \phi$, where $k_{\max}$ represents the maximum scaling parameter of data. This limit condition guarantees that as k approaches its maximum value, the reconstructed φ_k converges to the original data ϕ. This behavior is pivotal for the accuracy and fidelity of the generative process. Conversely, ϕ_0 simply follows the law of a tractable distribution.

Inter-scaling Interpolant. Since most of the data we aim to generate is discrete in nature, the (k_x, k_y, k_z) values in the K-amplitude decomposition are inherently defined on a lattice. Consequently, the derived scaling parameter k also takes discrete values. This discreteness implies that φ_k is originally defined only for discrete values of k. However, this discrete flow imposes a limitation: we cannot leverage the powerful flow-matching objective as the optimization framework, which requires taking derivatives with respect to continuous scaling step k.

To handle this issue, a straightforward approach is to extend φ_k to continuous k by intra-scaling interpolation. That is, we want a continuous flow Ψ_k, where $k \in [0, K]$ and satisfy $\Psi_k = \varphi_k$ for integer values of k. Let $t := k - \lfloor k \rfloor$ represent the continuous scaling step, where $\lfloor k \rfloor$ denotes the integer part of k. Then, the differentiable interpolation of Ψ_k is:

$$\Psi_k = \Psi_{\lfloor k \rfloor + t} = \mathcal{F}^{-1}\Bigg(\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}<\lfloor k \rfloor} \cdot \mathcal{F}\{\phi\}(k_x, k_y, k_z) + \Big(1 - \mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\geq\lfloor k \rfloor+1}\Big) \cdot \epsilon$$
$$+ \mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\in[\lfloor k \rfloor, \lfloor k \rfloor+1)} \cdot \big(\mu(t) \cdot \mathcal{F}\{\phi\}(k_x, k_y, k_z) + (1-\mu(t)) \cdot \epsilon\big)\Bigg), \quad (8)$$

where $\mu(t)$ is a bump function such that $\mu(0) = \mu(1) = 1$ and $\mu'(0) = -\mu'(1)$. The antisymmetric property of $\mu'(t)$ ensures that Ψ_k is differentiable from k for all $\mathbb{R}^+$, allowing the flow matching loss and other gradient-based optimization techniques. In Equation (8), we have three components: (1) $\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}<\lfloor k \rfloor}$ applies to the amplitude components up to the integer part of k; (2) $\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\geq\lfloor k \rfloor+1}$ applies noise padding to components beyond the next integer; (3) $\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\in[\lfloor k \rfloor, \lfloor k \rfloor+1)}$ performs linear interpolation of the intermediate amplitude components based on the current t.

Localized Vector Fields. Instead of directly modeling Ψ_k, we pivot our focus to its conditional gradient field, $\frac{d\Psi_k}{dk}$. By concentrating on the gradient field, we facilitate a dynamic view of how ϕ_k evolves with respect to k. To derive an analytical expression of $\frac{d\Psi_k}{dk}$ conditioned on a given instance pair of data and noise: (ϕ, ϵ), in what follows, we assume that $\mathcal{F}$ is a linear transform. Then, following Equation (8), we have the conditional vector field:

$$\frac{d\Psi_k}{dk}(\phi, \epsilon) = \mathcal{F}^{-1}\Big(\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\in[[k],[k]+1)} \cdot \mu'(t) \cdot (\epsilon - \mathcal{F}\{\phi\}(k_x, k_y, k_z))\Big), \quad (9)$$

for $k \in [\lfloor k \rfloor, \lfloor k \rfloor + 1)$ and $t = k - \lfloor k \rfloor$. Then, following Equation (6), the training objective of K-Flow is to learn the unconditional vector field in Equation (3) by the conditional flow matching:

$$\mathcal{L}_{\text{K-Flow}} := \mathbf{E}_{\phi_0} \int_0^K d\phi_0 \, dk \left\| \frac{d\Psi_k}{dk} - v_k(\Psi_k, \theta) \right\|^2. \quad (10)$$

By examining Equation (8) closely, we observe that the vector field is naturally localized around a subset of points in the K-amplitude space that satisfy $\sqrt{k_x^2 + k_y^2 + k_z^2} \in [\lfloor k \rfloor, \lfloor k \rfloor + 1)$. This localization means that the reconstruction at any given k primarily involves K-amplitude components within a narrow frequency band around k. Compared with the flow scheme in the pixel space, the K-amplitude in K-Flow reduces the optimization complexity by restricting the conditional vector field to be within a sub-manifold for each k. This sub-manifold may potentially be of low dimensionality, allowing for more focused updates and reducing the optimization space's dimensionality at each step. We will check how this localized conditional vector field affects the generation path in Appendix B.

We can further generalize the interpolation interval from $(\lfloor k \rfloor, \lfloor k \rfloor + 1)$ to (k_m, k_n), where k_m and k_n are two integers such that $k_m < k_n$. This adjustment broadens the range for intermediate amplitude components from $\sqrt{k_x^2 + k_y^2 + k_z^2} \in [\lfloor k \rfloor, \lfloor k \rfloor + 1)$ to $\sqrt{k_x^2 + k_y^2 + k_z^2} \in [k_m, k_n)$. For example, for our experiments, we partition the K-amplitude into two or three parts. See Appendix C.2 and Appendix C.3 for detailed implementations of these partitioning strategies.

3.2 EXAMPLES OF K-AMPLITUDE TRANSFORMATION

As we can see from Equation (1), all K-amplitude decompositions are achieved through expansion across a complete set of basis functions. However, the behavior of a K-amplitude decomposition (transform) can vary significantly depending on the choice of basis functions. Besides the Fourier transform introduced in Section 2, we provide two additional examples of K-amplitude decomposition: Wavelet transformation and PCA transformation. More details are in Appendix C.

Wavelet Transform. Wavelet decomposition (transform) deals with data that are not only scaling-localized but also spatially localized. The scaling parameter of wavelet transform is closely related to the notion of multi-resolution analysis [41], which provides a systematic way to decompose a signal into approximations and details at successively finer scales. This hierarchical decomposition is achieved through a set of scaling functions $\omega(x)$, and wavelet functions $\psi(x)$, which together serve as basis functions for the wavelet transformation. More precisely, in the wavelet transform, a signal $f(t)$ is expressed as a sum of scaled and translated versions of these basis functions times the corresponding coefficients c and d:

$$f(t) = \sum_j c_{k_0,j}\omega_{k_0,j}(t) + \sum_{k>k_0}\sum_j d_{k,j}\psi_{k,j}(t), \tag{11}$$

where $\omega_{k_0,j}(t)$ and $\psi_{k,j}(t)$ are the scaled and translated scaling and wavelet functions, respectively. The index j, which originally denotes the translation parameter, groups the basis within each fixed scaling parameter k naturally. Let $\phi_k := \sum_j d_{k,j}\psi_{k,j}$ for $k > k_0$ and $\phi_k := \sum_j c_{k_0,j}\omega_{k_0,j}$ for $k = k_0$, then eq. 11 is just one realization of K-amplitude decomposition. Concrete formulas for different families of wavelet bases, such as Daubechies (db), Meyer, and Haar, are in Appendix C.

In this article, we employ the discrete version of wavelet transform (DWT) as our K-amplitude transformation $\mathcal{F}$, which shares the linearity property with the Fourier transform with a bounded scaling parameter k, providing a structured yet flexible means of decomposing discrete data.

Date-dependent PCA Transform. Note that Fourier and wavelet decompositions are nonparametric k-amplitude decompositions that are independent of data. While these transformation methods are powerful, we also aim to find data-dependent decompositions that can capture common characteristic features specific to a given dataset. This motivation leads to principal component analysis (PCA), a technique widely used for the low-dimensional approximation of the data manifold [25]. Please consult appendix C for the K-amplitude realization of PCA transform.

K-amplitude Decomposition As A good Inductive Bias. From a data modeling perspective, it is valuable to study the statistics of data distribution across scalings, as defined by the specific K-amplitude decomposition we utilize. If the data distribution does not exhibit K-Flow scaling inhomogeneity, then all scalings should be treated equally, providing no justification for using a scaling-split generation path. As to latent data modeling, such as the latent space of an autoencoder, which is our main focus, we statistically analyze the mean norm of each scaling band across images in Figure 6. Obviously, we find that even in the compressed latent space, the mean norm of each scaling band decreases from low to high scalings. From the perspective of approximation error and model complexity, it is advantageous to allocate more refined sampling steps (or more model parameters) to lower scalings, as they contain more energy. On the other hand, pathological medical imaging data [10] may place more emphasis on the reconstruction of high-frequency components. In such cases, we need to allocate more computational resources to the high-scaling part of the K-Flow. Overall, our method can better capture significant features and maintain fidelity in the generated outputs.

3.3 PRACTICAL IMPLEMENTATION AND DISCUSSION

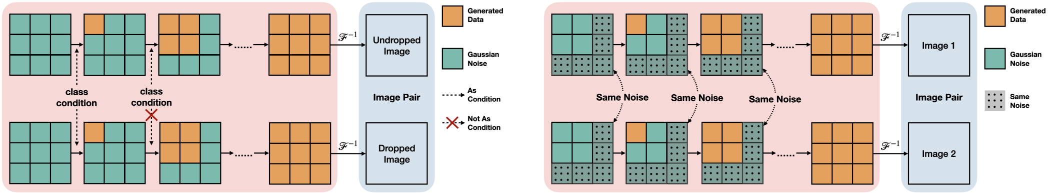
The overall structure of K-Flow is agnostic to the neural network architecture (for training the vector field), meaning that classical model architectures, such as U-net [55] and vision transformers [46], which are commonly used for training ordinary continuous normalizing flows or diffusion models, can be directly applied to K-Flow. This adaptability ensures that existing computational investments in these architectures can be effectively leveraged, providing a seamless transition to incorporating K-amplitude-based methods.

A Flexible Plug-In Version. To integrate our method into these existing models, we only introduce one targeted modification: replacing the time-embedding module with a K-amplitude-embedding module. Specifically, the time input of the time-embedding module in the diffusion transformer (or U-net) is substituted by the scaling parameter k. This substitution enables the K-Flow to leverage scaling information directly (especially the bump function), aligning with the principles of K-amplitude decomposition while preserving the original architecture's overall structure.

In addition to modifying this embedding module, to fully realize Equation (9), we provide several replaceable implementations of the bump functions in Appendix C for exploring the design space of our proposed K-Flow. Finally, we present the complete algorithm in Algorithm 1. For additional insights on the K-amplitude localization property and its implications for designing more efficient models, please refer to Appendix C.4.

4 EXPERIMENTS

We evaluate our K-Flow in multiple tasks, including unconditional image generation, class-conditioned image generation, and three ablation studies. The training algorithm is illustrated in Algorithm 1, and the implementation and hyperparameters are detailed in Appendix C.



(a) Controllable class-conditional generation. (b) Scaling-controllable generation (low scaling).

Figure 3: Pipelines of ablation study.

Table 1: Unconditional generation on CelebA-HQ.

Model	FID↓	Recall↑
CelebA-HQ 256		
K-Flow, Fourier-DiT L/2 (Ours)	5.11	0.47
K-Flow, Wave-DiT L/2 (Ours)	**4.99**	0.46
K-Flow, PCA-DiT L/2 (Ours)	5.19	0.48
LFM (ADM) [11]	5.82	0.42
LFM (DiT L/2) [11]	5.28	0.48
FM [36]	7.34	-
LDM [50]	5.11	**0.49**
LSGM [58]	7.22	-
WaveDiff [47]	5.94	0.37
DDGAN [61]	7.64	0.36
Score SDE [55]	7.23	-

Table 2: Class-conditional generation on ImageNet.

Model	FID↓	CDR↓	Recall↑
K-Flow, Wave-DiT L/2 (Ours)	17.8	-	0.56
+ cfg=1.5	4.49	-	0.44
K-Flow, Fourier-DiT L/2 (Ours)	13.5	-	0.57
+ cfg=1.5	**2.77**	**1.49**	0.45
LFM (DiT L/2)	14.0	-	0.56
+ cfg=1.5	2.78	3.25	0.42
LDM-8 [50]	15.51	-	**0.63**
LDM-8-G	7.76	-	0.35
DiT-B/2 [46]	43.47	-	-

4.1 IMAGE UNCONDITIONAL GENERATION

The first task is to generate random samples after fitting a target data distribution, which is typically concentrated around a low-dimensional sub-manifold within the ambient space.

Dataset and Metrics. We conduct experiments on the CelebA-HQ [27] dataset with a resolution of 256×256. To evaluate the performance of our proposed method, we employ two metrics: the Fréchet Inception Distance (FID) [22], which evaluates the quality by measuring the statistical similarity between generated and real images, and Recall [30], which measures the generation diversity.

Results. Table 1 summarizes the comparison between our proposed K-Flow model and other generative models. For a fair comparison, both the baseline ordinary flow matching [11] and our K-Flow flow utilize the same VAE's latent from [50] and the Diffusion Transformer with the same size (*e.g.*, DIT L/2 [46]) as the backbone model. We can observe that (1) K-Flow achieves the best performance in FID, especially w/ the db6-based wavelet K-Flow. (2) Although the latent diffusion model [50] gets the highest score in Recall (diversity), the Fourier and PCA-based K-Flow is comparable with the ordinary latent flow matching.

4.2 IMAGE CLASS-CONDITIONAL GENERATION

In this section, we explore how K-amplitude decomposition behaves when the generation path is conditioned on class labels, where the class label (*e.g.*, dog, cat, fish, etc) delegates the low-scaling information of each image. This investigation could potentially pave the way for multi-modal scaling control, where different scaling components are influenced by specific caption information. We list the detailed class-conditional generation algorithm in Appendix D.

Dataset and Metric. We use ImageNet as the middle-size conditional generation dataset [12]. Beyond evaluating the unconditional FID for the ImageNet dataset, we are interested in studying how the class control interacts with scaling generation in a quantitative manner. Details in Appendix E.

Results. The results are presented in Table 2. Our primary focus for the FID metric is the classifier-free guidance inference method applied to flow matching models. The data indicates that K-Flow achieves results comparable to LFM. In terms of the recall metric, which assesses the diversity of the generated distribution, our model outperforms the standard LFM. This improvement may be attributed to the fact that the inference path of K-Flow includes a greater number of dimensions during the low-scaling


(a) Low scaling. (b) High scaling.

Figure 4: Results of scaling-controllable generation.

period, as discussed in Appendix B.1. Given that the inference path of K-Flow accommodates the K-amplitude scalings, we anticipate that omitting the class label (a low-scaling caption) in the high-scaling segments will not substantially impact FID. Our observations confirm this expectation: the conditional discrimination ratio (CDR, defined in appendix E) of our model is close to one, indicating a balanced performance. In contrast, the CDR of the conventional LFM is significantly higher, suggesting a discrepancy in performance under these conditions. For the qualitative analysis, see Section 4.3. This preliminary exploration suggests that our proposed K-Flow has the potential to allocate computational resources more efficiently by leveraging the correlation between K scaling and captions.

4.3 CONTROLLABLE CLASS-CONDITIONAL GENERATION

The latent flow matching model can implicitly learn low- and high-resolution features [11], but the boundary between each resolution is vague, and we cannot explicitly determine which timestep in the inference process corresponds to a specific resolution or frequency. In comparison, our proposed wavelet-based K-Flow enables finer-grained controllable generation. As shown in Figures 3a and 5, when we drop the class conditions during the last 70% scaling steps of the inference process, K-Flow can effectively preserve high-frequency details, whereas the ordinary latent flow tends to blur the entire image.

Figure 5: Results of controllable class generation.

4.4 SCALING-CONTROLLABLE GENERATION

Preserving High Scaling, Modifying Low Scaling. This scaling-controllable generation pipeline is illustrated in Figure 3b. It involves sampling multiple images while ensuring that the noise in the high-scaling components remains consistent across all samples. In scaling-controllable image generation, the goal is to maintain consistency in the high-scaling details while allowing variations in the low-scaling context among the generated images, thus this allows K-Flow to achieve unsupervised steerability in a finetuning-free manner. The algorithm is in Appendix D.

The results on CelebA are presented in Figure 4a, where we apply a pretrained Daubechies wavelet (db6-based) K-Flow. It can be observed that facial details, such as eyes, smiles, noses, and eyebrows, remain consistent within each group of images. In contrast, the low-scaling components, including background, gender, age, and hairstyle, vary across the images within the same group.

Preserving Low Scaling, Modifying High Scaling. We need to highlight that in K-Flow, when modeling the flow from lower to higher scales, the noise at higher scales is used to predict the velocity at the lower scale. This is determined by the nature of ODE flow. To this end, we conduct an ablation study by reversing the scaling direction in the Daubechies wavelet K-Flow, and the pipeline is illustrated in Figure 14 (Appendix E). In such a reversed setup, we keep the low-scaling part the same noise, while gradually denoising the high-scaling part.

The results are listed in Figure 4b. According to the six pairs of results, we can observe that the low-scaling part like the background of the image and the gender and color of the people stay the same, while the high-resolution details of facial expressions and outlook vary within each pair.

REFERENCES

[1] Patrice Abry, Paulo Gonçalvès, and Patrick Flandrin. *Wavelets, spectrum analysis and 1/f processes*, pp. 15–29. Springer New York, New York, NY, 1995. ISBN 978-1-4612-2544-7. doi: 10.1007/978-1-4612-2544-7_2. URL https://doi.org/10.1007/978-1-4612-2544-7_2. 2

[2] Ali Naci Akansu and Richard A. Haddad. Multiresolution signal decomposition: Transforms, subbands, and wavelets. 1992. URL https://api.semanticscholar.org/CorpusID:60630002. 21

[3] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022. 1, 3

[4] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023. 4

[5] Yuval Atzmon, Maciej Bala, Yogesh Balaji, Tiffany Cai, Yin Cui, Jiaojiao Fan, Yunhao Ge, Siddharth Gururani, Jacob Huffman, Ronald Isaac, et al. Edify image: High-quality image generation with pixel space laplacian diffusion models. *arXiv preprint arXiv:2411.07126*, 2024. 13, 16, 18

[6] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pp. 671–679. Elsevier, 1987. 13, 18

[7] Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pp. 33–40. PMLR, 2005. 1

[8] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 16

[9] Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. Deconstructing denoising diffusion models for self-supervised learning. *arXiv preprint arXiv:2401.14404*, 2024. 17

[10] Yuetan Chu, Yilan Zhang, Zhongyi Han, Changchun Yang, Longxi Zhou, Gongning Luo, and Xin Gao. Improving representation of high-frequency components for medical foundation models. *arXiv preprint arXiv:2407.14651*, 2024. 6

[11] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023. 7, 8

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 7

[13] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. *Advances in neural information processing systems*, 28, 2015. 13

[14] Weitao Du, He Zhang, Tao Yang, and Yuanqi Du. A flexible diffusion model. In *International Conference on Machine Learning*, pp. 8678–8696. PMLR, 2023. 17

[15] Weinan Ee. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5:1–11, 02 2017. doi: 10.1007/s40304-017-0103-z. 17

[16] Carlos Esteves, Mohammed Suhail, and Ameesh Makadia. Spectral image tokenizer. *arXiv preprint arXiv:2412.09607*, 2024. 14

[17] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12):2379–2394, 12 1987. doi: 10.1364/JOSAA.4.002379. URL https://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-12-2379. 2

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 13, 16

[19] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Miguel Angel Bautista, and Josh Susskind. f-dm: A multi-stage diffusion model via progressive signal transformation. *arXiv preprint arXiv:2210.04955*, 2022. 13

[20] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010. 1

[21] Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 71(1): 38–53, 1911. 26

[22] 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. 7

[23] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 1

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 4, 13

[25] Alan Julian Izenman. Introduction to manifold learning. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(5):439–446, 2012. 6

[26] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024. 17

[27] Tero Karras. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 7

[28] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 18

[29] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1, 13, 16

[30] 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. 7

[31] Hugo Lavenant and Filippo Santambrogio. The flow map of the fokker–planck equation does not provide optimal transport. *Applied Mathematics Letters*, 133:108225, 2022. 16

[32] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 1

[33] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022. 13

[34] Jiarui Lei, Xiaobo Hu, Yue Wang, and Dong Liu. Pyramidflow: High-resolution defect contrastive localization using pyramid normalizing flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14143–14152, 2023. 16

[35] Zhengqi Li, Richard Tucker, Noah Snavely, and Aleksander Holynski. Generative image dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24142–24153, 2024. 16

[36] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 1, 3, 4, 7, 14

[37] Shengchao Liu, Hongyu Guo, and Jian Tang. Molecular geometry pretraining with se (3)-invariant denoising distance matching. *arXiv preprint arXiv:2206.13602*, 2022. 17

[38] Shengchao Liu, Yanjing Li, Zhuoxinran Li, Zhiling Zheng, Chenru Duan, Zhi-Ming Ma, Omar Yaghi, Animashree Anandkumar, Christian Borgs, Jennifer Chayes, et al. Symmetry-informed geometric representation for molecules, proteins, and crystalline materials. *Advances in neural information processing systems*, 36, 2024. 17

[39] Sifei Liu, Shalini De Mello, and Jan Kautz. Cosae: Learnable fourier series for image restoration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 17

[40] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 1, 3, 4

[41] Stephane G Mallat. Multiresolution approximations and wavelet orthonormal bases of $l^2(r)$. *Transactions of the American Mathematical Society*, 315(1):69–87, 1989. 5

[42] Wael Mattar, Idan Levy, Nir Sharon, and Shai Dekel. Wavelets are all you need for autoregressive image generation. *arXiv preprint arXiv:2406.19997*, 2024. 17

[43] Yves Meyer. Ondelettes et opérateurs. *I: Ondelettes*, 1990. 26

[44] Yves Meyer. *Wavelets and operators: volume 1*. Number 37. Cambridge university press, 1992. 20

[45] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. 16

[46] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. 6, 7, 22

[47] Hao Phung, Quan Dao, and Anh Tran. Wavelet diffusion models are fast and scalable image generators. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10199–10208, 2023. 7, 14, 17

[48] Alec Radford. Improving language understanding by generative pre-training. 2018. 1

[49] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 13

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022. 7, 22

[51] Dohoon Ryu and Jong Chul Ye. Pyramidal denoising diffusion probabilistic models. *arXiv preprint arXiv:2208.01864*, 2022. 13

[52] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022. 13

[53] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1, 13

[54] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021. 1

[55] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 6, 7

[56] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024. 13, 17, 18

[57] Hoai-Chau Tran, Duy MH Nguyen, Duy M Nguyen, Trung-Tin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Y Zou, Binh T Nguyen, and Mathias Niepert. Accelerating transformers with spectrum-preserving token merging. *arXiv preprint arXiv:2405.16148*, 2024. 14

[58] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Neural Information Processing Systems (NeurIPS)*, 2021. 7

[59] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 1, 13, 16

[60] Yair Weiss and William T Freeman. What makes a good model of natural images? In *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8. IEEE, 2007. 19

[61] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021. 7

[62] Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023. 14

[63] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7777–7786, 2024. 15, 16

A RELATED WORK

There have been multiple research lines on studying generative modeling, especially in terms of multi-scale modeling. In this work, we would like to summarize them as the following three venues.

A.1 MULTI-SCALE IN PIXEL RESOLUTION, FLOW AND DIFFUSION

Laplacian Pyramid and Laplacian Operator. In mathematics, the Laplacian operator computes the second derivative of a function, emphasizing regions with significant intensity changes, such as edges or high-frequency details. Similarly, the Laplacian Pyramid [6] decomposes an image into multiple scales, extracting the low-frequency components (smooth regions) through downsampling. The high-frequency details, such as edges and textures, are modeled as the residuals between adjacent resolution layers. The primary objective of the Laplacian Pyramid is to represent these residuals across scales in a hierarchical fashion.

LAPGAN (Laplacian Generative Adversarial Networks) [13] adopts the Laplacian pyramid idea into the generative adversarial network (GAN) framework [18]. By focusing on learning residuals between successive levels of resolution, it effectively generates high-quality super-resolution images.

SR3 (Super-Resolution via Repeated Refinement) [52] leverages DDPM (Denoising Diffusion Probabilistic Models) [24] and DSM (Denoising Score Matching) [53, 59] for high-resolution image generation. Specifically, SR3 enhances low-resolution images to high-resolution by utilizing multiple cascaded conditional diffusion models. In this framework, the low-resolution images serve as conditions, and the model's aim is to predict the corresponding high-resolution images as outputs.

PDDPM (Pyramidal Denoising Diffusion Probabilistic Models) [51] is a follow-up work of SR3, and it improves the model by only modeling one score network. The key attribute to enable this is by adding the fractional position of each pixel to the score network, and such fractional position information can be naturally generalized to different resolutions.

f-DM [19] is developed concurrently with PDDPM and shares the approach of utilizing only one diffusion model. It distinguishes itself by explicitly applying a sequence of transformations to the data and emphasizing a resolution-agnostic signal-to-noise ratio for noise scaling within its diffusion model design.

Edify Image [5] is a state-of-the-art model capable of generating photorealistic, high-resolution images from textual prompts [5]. It operates as a cascaded pixel-space diffusion model. To enhance its functionality, Edify Image employs a downsampling process that extracts low-frequency components and creates three distinct resolution levels, ranging from low to high frequency, with the original image representing the highest frequency level. Another key innovation of Edify Image is its meticulously crafted training and sampling strategies at different resolutions, utilizing attenuated noise schedules.

A.2 MULTI-SCALE IN PIXEL RESOLUTION, VAE AND AR

VQ-VAE2 (Vector Quantized VAE 2) [49] enforces a two-layer hierarchical structure, where the top layer captures global features such as object shapes and geometry, while the bottom layer focuses on local details like texture. It models data density within the variational autoencoder (VAE) framework[29] and incorporates an autoregressive (AR) module to enhance the prior for improved generative performance.

RQ-VAE (Residual-Quantized VAE) [33] integrates recursive quantization into the VAE framework. It constructs a representation by aggregating information across D layers, where the first layer provides a code embedding closely aligned with the encoded representation, and each subsequent layer refines this by reducing the quantization error from the previous layer. By stacking D layers, the accumulated quantization error is minimized, enabling RQ-VAE to offer a coarse-to-fine-grained approach to modeling. For modeling, the general pipeline follows the VAE framework, while each latent code is decomposed into D layers and is predicted in an autoregressive manner.

VAR (Visual AutoRegressive) [56] introduces a novel paradigm for density estimation by decomposing images into multiple resolutions across various scales. This approach is inspired by the hierarchical nature of human perception, where images are interpreted progressively from global structures to finer details. Leveraging this concept, VAR models the entire image in a coarse-to-fine manner, adhering to the principles of multi-scale hierarchical representation.

A.3 MULTI-SCALE IN FREQUENCY, AR AND DIFFUSION

WaveDiff (Wavelet Diffusion) [47] leverages the discrete wavelet transform to shift the entire diffusion process into the wavelet spectrum. Its primary objective is to reduce model complexity by operating in the transformed spectrum space instead of the pixel domain.

PiToMe (Protect Informative Tokens before Merging) [57] is a token merging method designed to balance efficiency and information retention. PiToMe identifies large clusters of similar tokens as high-energy regions, making them suitable candidates for merging, while smaller, more unique, and isolated clusters are treated as low-energy and preserved. By interpreting attention over sequences as a fully connected graph of tokens, PiToMe leverages spectral graph theory to demonstrate its ability to preserve critical information.

SIT (Spectral Image Tokenizer) [16] is a parallel work to ours that processes the spectral coefficients of input patches (image tokens) obtained through a discrete wavelet transform. Motivated by the spectral properties of natural images, SIT focuses on effectively capturing the high-frequency components of images. Furthermore, it introduces a scale-wise attention mechanism, referred to as scale-causal self-attention, which is designed to improve the model's expressiveness across multiple scales.

B DISCUSSION

B.1 FROM CONDITIONAL TO UNCONDITIONAL PATH IN K-FLOW

In Section 3, our frequency-localized path is defined at the conditional level ($\frac{d\Psi_k}{dk}(\phi, \epsilon)$) , and it is only related to the unconditional vector field ($v_k(\Psi_k, \theta)$ in eq. (10)) through the equivalence of conditional flow matching and unconditional flow matching at the loss level [36]. In this section, we try to study the splitting property of the unconditional K-amplitude vector field.

By the K-amplitude decomposition, the transformed data probability p_{data} satisfies the telescoping property:

$$p_{data} = p(k_0)p(k_1|k_0)\dots p(k_{max}|k_{max}-1,\dots,k_0), \tag{12}$$

with k_0 and k_{max} denoting the lowest and highest scaling. Then, according to the definition of our proposed K-amplitude flow Ψ_k, the interpolated probability at scaling step t is also localized:

$$p_t(\cdot) = p(k_0)\cdots p_t(\cdot|\lfloor k\rfloor,\dots,k_0)p_\epsilon(\lfloor k\rfloor+1)\cdots p_\epsilon(k_{max}|k_{max}-1,\dots,k_0), \tag{13}$$

where p_ϵ denotes the distribution of the initial noise and $t \in [\lfloor k\rfloor, \lfloor k\rfloor+1)$. Combining Equation (13), the localization property of the bump function and Lemma 1 of [62], the unconditional vector field has an explicit form: $v_t(\Psi_k) = a_t \cdot \Psi_k + b_t \nabla \log p_t(\Psi_k)$, where a_t and b_t are hyper-parameters determined by the bump function we choose.

Noise Splitting A key characteristic of flow models is their deterministic nature after the initial noise sampling. Specifically, once the initial noise is sampled, the flow follows a fixed path to generate the final data sample. According to Equation (13), during scaling step t: (1) the scaling components below $\lfloor k\rfloor$ remain unchanged; (2) the scaling components above $\lfloor k\rfloor$ remain unchanged; (3) The distribution of higher scaling components maintains the same characteristics as their initial noise distribution.

By these observations, we now investigate how segmented initial noise in the K-Flow space influences the final output of the K-Flow flow. Suppose we discretize scaling parameter k into two parts: $\mathcal{F}\{\Psi_k\} = \{\phi_{\text{low}}(k), \phi_{\text{high}}(k)\}$. When flowing along the low-scaling component, the vector field v_k can be re-expressed in a conditional form:

$$v_k(\Psi_k) = v_k(\phi_{\text{low}}(k), c) \tag{14}$$

where constant c represents the (static) initial noise for the high-scaling part. This noise-conditioned property in the k-amplitude domain leads us to explore whether fixing the high-scaling noise and

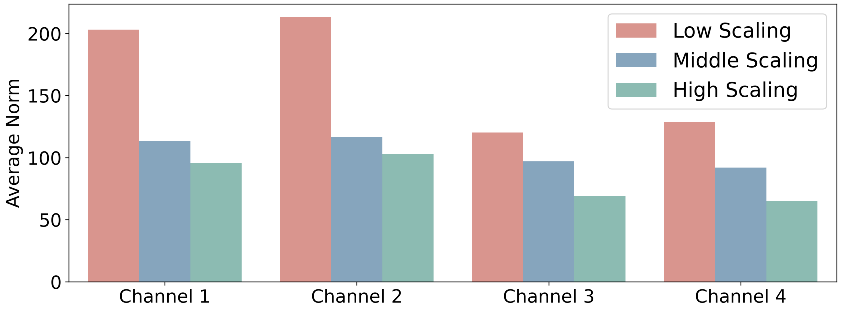


Figure 6: On the low-scaling hypothesis. The graph illustrates the relative norm distribution for each scaling component as defined by the wavelet decomposition in the latent space. It can be observed that the low-scaling component exhibits a significantly higher norm (energy), nearly twice that of the high-scaling component.

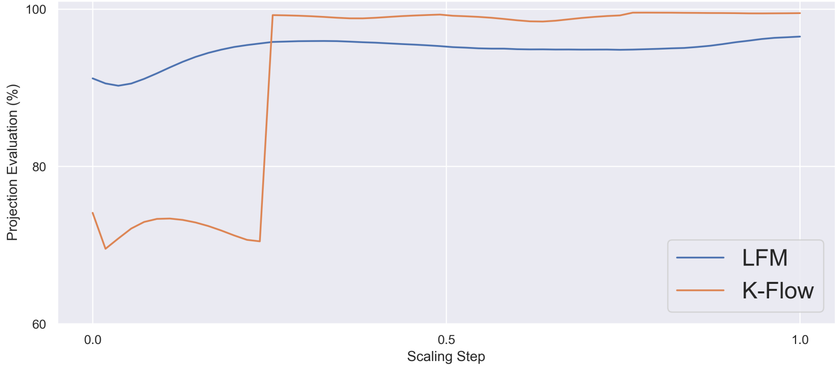


Figure 7: Projection Error Comparison with Different Models. The graph illustrates the PCA projection errors of two models throughout the entire flow process, with distinct segments marked by dashed lines. The red and blue lines represent the original flow and the k-Amplitude flow with two amplitude components, respectively. The projection error is quantified by the reconstruction error for each generation step from the PCA compression, using the first two principal components. Owing to the scaling-aware nature of our flow, the low-amplitude portion (the initial part of the curve) resides in a relatively high-dimensional space, resulting in higher projection errors for the two-dimensional PCA projection.

altering the low-scaling noise allows for unsupervised editing of relative low-scaling semantics in an image. Indeed, we observed this phenomenon, the qualitative results will be discussed in section 4.4.

From Figure 4a, we observe that a targeted common high-scaling initial noise guides our K-Flow flow toward generating human faces with similar detail but varying low-level content. See the experiment section for a more detailed analysis.

B.2 THE EFFECT OF SCALING STEP k FOR IMAGE RECONSTRUCTION

K-Flow's ability to leverage the low-dimensional structure of data is primarily enabled by its K-Flow localization property. This enables a strategic path through low-dimensional spaces, which can be directly compared with the generation path of conventional flow models. In our model, this path incorporates an explicit frequency hierarchy, which hypothesizes that the low-frequency components - concentrated in the earlier stages of the model - may share more dimensions in common, particularly from a semantic perspective, than the high-frequency components positioned later in the generative process. Conversely, an ordinary flow model may exhibit a more uniform distribution of dimensionality across the entire generative path.

Motivated by this hypothesis, we conduct a case study using PCA to approximate the dimension of the generation trajectory $\{\Psi_k\}_{k=k_0}^{k_{max}}$. As illustrated in Figure 7, we measure how closely the dimension of the generation path aligns with a two-dimensional subspace spanned by the first two components of the model's PCA decomposition, denoted by $\{\tilde{\Psi}_k\}_{k=k_0}^{k_{max}}$. Inspired by [63], the reconstruction ratio is defined by $1 - \|\Psi_k - \tilde{\Psi}_k\|_2 / \|\Psi_k\|_2$. In other words, a higher value of the reconstruction ratio

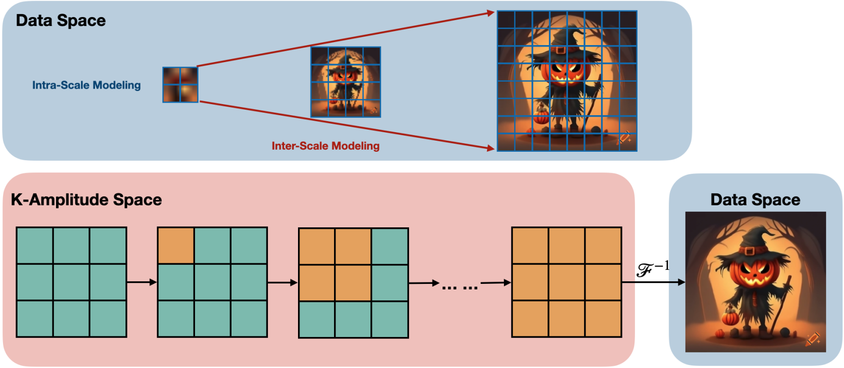


Figure 8: Comparison of multi-scale modeling: pixel data space and K-Amplitude space.

indicates that the model's dimension is closer to two. Therefore, the trend of the error curve with respect to the scaling parameter k reveals a distinct separation in the effective dimension between low- and high-scaling components. Evidently, the low-scaling segments display more semantic consistency and thus, occupy a larger dimensional space, whereas the high-scaling segments converge to a more confined or lower-dimensional structure.

It is important to note that this exploration into the dimensionality of generative paths is practically meaningful. Previous study [63] has shown that the effectiveness of distilling a generative model with fewer steps from a pre-trained diffusion model theoretically depends on the model's dimensionality at each step, as informed by the high-dimensional Mean Value Theorem. The observations from Figure 7 provide empirical support for this concept. Specifically, the ability of K-Flow to maintain a lower-dimensional structure in high-scaling components suggests a promising approach for fast sampling distillation methods.

B.3 RELATED WORK DISCUSSION

The field of generative modeling has seen significant advancements in recent years, driven by a variety of frameworks, including adversarial generative networks (GAN) [18], variable autoencoders (VAE) [29], and normalizing flows [45]. In this work, we focus on continuous normalizing flow generative models [8], with particular emphasis on the conditional flow matching training scheme, which originates from the denoising score matching training framework [59].

Both diffusion models and continuous flow matching models aim to lower the complexity of directly optimizing the log-likelihood of data by introducing an additional stochastic path. However, as proved in [31], the canonical path for diffusion models and rectified flows is not optimal. This realization motivates our introduction of frequency decomposition as a key design element in generative models.

By breaking down the formula of our K-Flow vector field with respect to the scaling parameter k, we can summarize three successful factors as general principles for (frequency) scaling modeling.

- A good K-amplitude decomposition can leverage the problem's inherent biases towards certain scaling bands. For generative tasks, it is crucial that all K-Flow bands are effectively modeled to ensure the generation of high-quality, controllable outputs. In addition, the computational resources required may vary between different scales, thus necessitating careful consideration of resource allocation.
- Modeling within each scaling component, which is formulated in our K-Flow-localized vector fields.
- Modeling bridges along different scalings, which is achieved through our flow ODE and the (time) K-Flow embedding block for the U-Net or DIT architecture.

This approach to inter- and intra-modeling for K-amplitude is also applicable to scenarios emphasizing certain frequencies or scalings. For instance, [35] enhanced oscillatory motion control in video generation by discarding the high-frequency component of the Fourier decomposition. As discussed in Section 3, the scaling parameter of spatially localized wavelet (multi-resolution) decomposition is closely linked to image resolution. Notable contributions in this domain include [5] and [34], which introduced a multi-stage resolution for fine-grained editing, and [26], which concentrated on efficient video generation.

In related research on auto-regressive modeling, [42] presented wavelets as an effective auto-regressive unit, while [56] focused on the scale as a key element for image auto-regression. A significant example is [47], which transitioned the latent space from pixel to wavelet space for generative models using wavelet diffusion. However, their method employed the same conditional noising schedule for score matching as traditional diffusion models. In contrast to their approach, our proposed K-Flow integrates wavelet decomposition as a multi-channel module within the neural network architecture for training diffusion models. Additionally, our work extends the notion of wavelet space to the more general K-amplitude space.

We also want to highlight another research line that has recently caught the attention is the auto-regressive over the pixel space for image generation. One classic work is VAR [56]. It introduces a hierarchical density estimation paradigm that models images in a coarse-to-fine manner across multiple resolutions and models the data distribution in an auto-regressive manner. In contrast, our proposed K-Flow integrates the flow paradigm for density estimation and leverages the K-amplitude space as a stronger inductive bias, as illustrated in Figure 8.

Summary. In summary, K-Flow is a more general framework, with its three key factors potentially benefiting generation-related tasks like super-resolution and multi-resolution editing. For example, [39] utilized a learnable Fourier transform to construct a harmonic module in the bottleneck layer of an autoencoder. We provide a comprehensive list of related works in Appendix A.

B.4 CONNECTING K-FLOW WITH SSL REPRESENTATION AND GENERATION

From the above discussion, we have seen how pretrained vision models leverage the sparsity and locality of natural data in various K-amplitude domains for perception and generation-based tasks. In the realm of unsupervised learning, [9, 37, 38] explore whether generative-based representations, particularly those derived from denoising diffusion models, can achieve parity with contrastive-based representation learning methods for downstream tasks. A key observation from their findings [9], which aligns with our approach of employing K-amplitude decomposition (the PCA instance), is the revelation that the most powerful representations are obtained through denoising within a latent space, such as the compressed PCA space. Another merit of PCA is that denoising along the PCA directions can achieve faster convergence for denoising, which is revealed in [14].

To transition from unsupervised representation learning to real data generation, incorporating all K-amplitude scalings is essential. Rather than compressing or amplifying specific scaling bandwidths, generative tasks require novel organization or ordering of all frequencies. Besides our flow-based frequency generation approach, [56] connects different scales (which can be interpreted as the wavelet K-amplitudes) using residual connections, with an auto-regressive training objective. Residual connections, as a discretization of ordinary differential equations (ODEs) proposed in [15], suggest that [56]'s approach can be seen as a special discretization of our K-Flow with a flexible flow matching training objective.

Table 3: Comparison among PCA, contrastive, and generative SSL.

	Basis Learning	Reconstruction Learning
PCA SSL	Non-parameterized, Determined By Data	Parameterized
Contrastive SSL	Parameterized	N/A
Generative SSL	Parameterized	Parameterized

C METHOD DETAILS

C.1 SIX PROPERTIES OF K-FLOW

(a) K-Flow provides **a first-principle way to organize the scaling** k. Unlike perception-based computer vision tasks, which often favor certain scaling (frequency) bands, a K-amplitude based generative model strives for an optimal organization of all scalings to ensure that the final generated sample is of high fidelity. By constructing K-amplitude scaling-based vector fields, the integrated flow naturally incorporates all scaling information, and the conditional flow matching training objective provides a perfect trade-off of accuracy-efficiency inside localized scalings. We will also demonstrate how different discretizations of K-Flow with related works, highlighting the connections and integrations with existing methods in the field.

(b) K-Flow enables **multi-scale modeling in the K-amplitude space**. Compared to the original data space, such as the pixel space in images, the K-amplitude space provides a more natural perspective for defining and analyzing multi-scale information, namely, K-amplitude decomposition empowers K-Flow for effective multi-scale modeling. By decomposing the feature representation into multiple scaling components in the K-amplitude space, K-Flow associates each scaling with an amplitude. Higher values of K-amplitude correspond to higher-frequency information, capturing fine-grained details, while lower values encode lower-frequency information, representing more coarse-grained features. Let us take the image for illustration. Images inherently exhibit a hierarchical structure, with information distributed across various resolution levels. Low-resolution components capture global shapes and background information, while high-resolution components encode fine details like textures, often sparse and localized. By projecting these components into the K-amplitude space, K-Flow captures such hierarchical information effectively and naturally, enabling precise modeling of the interplay between scales.

(c) K-Flow supports **a well-defined scale along with energy**. The amplitude is also used to reflect the *energy* level at each scale of the data. In physics, it is proportional to the square of the amplitude. In comparison, for the modeling on the original data space, though we can inject application-specific inductive bias, such as multiple pixel resolutions for images, they do not possess a natural energy concept.

(d) K-Flow interprets **scaling as time**. From elucidating the design space of the traditional flow matching perspective, K-Flow re-defines the artificial time variable (or the signal-to-noise ratio variable proposed in [28]) as the ordering index of frequency space. In this context, the artificial time variable effectively controls the traversal through different levels of a general notion of frequency decompositions, scaling each frequency component appropriately. This perspective aligns with the concept of renormalization in physical systems, where behavior across scales is systematically related.

(e) K-Flow supports the **fusion of intra-scaling and inter-scaling modeling.** K-Flow flows across scaling as time, and namely, K-Flow naturally merges the intra- and inter-scaling during the flow process. Thus the key module turns to the smooth interpolant, as will be introduced in Section 3. This is in comparison with existing works on multi-modal modeling [5, 6, 56], where the special design of the intra-scaling and inter-scaling is required.

(f) K-Flow supports **explicit steerability**. The flow process across scales enables K-Flow to control the information learned at various hierarchical levels. This, in turn, allows finer-grained control of the generative modeling, facilitating more precise and customizable outputs. By understanding and leveraging K-Flow's steerability, its utility can be significantly enhanced across diverse domains, including Artificial Intelligence-Generated Content (AIGC), AI-driven scientific discovery, and the safe, responsible development of AI technologies.

C.2 METHOD DETAILS

C.2.1 REMARK ON K-AMPLITUDE DECOMPOSITION

Fourier transform We have shown how to build the K-Amplitude scaling through the Fourier space in Section 2.2. In the discrete setting, the Fourier transform is realized by basis functions of the form $W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$, where N is the length of the sequential data. An effective K-amplitude decomposition exploits this structure by aligning with the inherent hierarchical structure of the data manifold. For example, if most of the energy or amplitudes are concentrated in the low-scaling range, the generative capability of the flow can be enhanced by allocating more steps or resources to these low frequencies. Conversely, fewer steps can be allocated to high frequencies that carry minimal mass or information. For the Fourier transform, this tendency is evident in the analysis of natural images, which often exhibit the celebrated $1/f$ spectrum phenomenon [60]. This phenomenon suggests that energy diminishes with increasing scaling, meaning that low-scaling components hold the majority of the signal's information content.

PCA transform as a K-amplitude decomposition From the K-amplitude perspective, PCA is an eigen-decomposition obtained by the data covariance matrix. The covariance matrix is given by:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}_{\text{centered}}^{\top}\mathbf{X}_{\text{centered}},$$

where $\mathbf{X}_{\text{centered}} = \mathbf{X} - \overline{\mathbf{X}}$ is the centered data matrix. In this context, the principal components reveal the relative importance of each transformed direction. To translate PCA into a K-amplitude decomposition, we define the k scaling parameter as the relative order of the principal components. For implementation, we utilize the eigenvalue decomposition of $\mathbf{C}$ for PCA, and the eigenvalues in their descending ordering define the k scaling parameter.

C.2.2 IMPLEMENTATION DETAILS OF K-FLOW VECTOR FIELD

Scaling Discretization In the main text, we assume, by default, that the scaling parameter k takes integer values: $k \in \{0, 1, 2, \ldots, k_{\text{max}}\}$. Thus, the differentiable vector field v_k for continuous k is defined by interpolating between $\lfloor k \rfloor$ and $\lfloor k \rfloor + 1$.

We now extend this concept to a more general setting where k may take a limited set of integer values within the range from 0 to k_{max}. Suppose k_m and k_n represent two specific integer values for k. We demonstrate how to extend k continuously within the connected interval $[k_m, k_n)$. Let $t := k - k_m$. The differentiable version of ϕ_k is then expressed as:

$$\begin{aligned}\Psi_{k_m+t} =& \mathcal{F}^{-1}\Bigg(\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}<k_m} \cdot \mathcal{F}\{\phi\}(k_x, k_y, k_z) + \Big(1 - \mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\geq k_n}\Big) \cdot \epsilon \\ &+ \mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\in[k_m,k_n)} \cdot \big(\mu(t) \cdot \mathcal{F}\{\phi\}(k_x, k_y, k_z) + (1-\mu(t)) \cdot \epsilon\big)\Bigg),\end{aligned} \tag{15}$$

where $\mu(t)$ is a bump function fulfilling $\mu(0) = \mu(k_n - k_m) = 1$ and $\mu'(0) = -\mu'(k_n - k_m)$.

Replacing the Fourier transform with the general K-amplitude decomposition, the K-amplitude flow is expressed in its general form as follows:

$$\begin{aligned}\Psi_{k_m+t} =& \mathcal{F}^{-1}\Bigg(\mathbb{I}_{k<k_m} \cdot \mathcal{F}\{\phi\}(k) + \Big(1 - \mathbb{I}_{\sqrt{k}\geq k_n}\Big) \cdot \epsilon \\ &+ \mathbb{I}_{k\in[k_m,k_n)} \cdot \big(\mu(t) \cdot \mathcal{F}(\{\phi\}k) + (1-\mu(t)) \cdot \epsilon\big)\Bigg),\end{aligned} \tag{16}$$

where $\mathcal{F}\{\phi\}(k)$ is defined in the main text.

Experimental Implementation In this paper's experiments, particularly in the Fourier and PCA versions of the K-Flow flow, we restrict the discrete values of k to $\{0, \frac{k_{\text{max}}}{2}, k_{\text{max}}\}$, with k_{max} determined by resolution. We then extend k continuously using Equation 15.

Bump Function We propose two types of bump functions: 1. Hard bump; 2. Soft bump. The **hard bump function** $\mu : [0, 1] \to \mathbf{R}^+$ satisfies the specific endpoint properties:

$$\mu(0) = \mu(1) = 1 \quad \text{and} \quad \mu'(0) = -\mu'(1). \tag{17}$$

Inspired by spline functions, such bump functions can be constructed using polynomials. For example, a quartic form used in our experiments is given by:

$$\mu(t) = 1 - 3t^2 + 2t^3. \tag{18}$$

For more examples, readers can explore modifications of the connection functions used in Meyer wavelets [44].

In this paper, we utilize hard bump functions for constructing K-Flow flows with scaling discretization exceeding one components.

Soft Localization with Soft Bump Function Consider that the scaling parameter is discretized to take values in an increasing sequence $\{k_i\}_{i=0}^n$. Consequently, the continuous k lies in the interval $k \in [k_0, k_n]$. Define

$$\psi_i := \mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2} \in [k_i, k_{i+1})}.$$

These ψ_i form a partition of unity for the K-Amplitude basis. The derivative of the soft bump function μ_i' is defined for each scaling band ψ_i, expressed as:

$$\mu_i'(k; a_i, b) = \begin{cases} c \cdot \left(1 - \left(\frac{k-a_i}{b}\right)^2\right)^n, & \text{if } |k - a_i| < b, \\ 0, & \text{if } |k - a_i| \geq b, \end{cases} \tag{19}$$

where $a_i = \frac{k_i + k_{i+1}}{2}$ and c is the normalization constant ensuring that the integral of the function over its compact support is 1. Note that hyper-parameter $b \leq k_n - k_0$ dictates the width or support region of the bump, while the degree n measures the sharpness of the bump. We retain b and n as hyperparameters. The bump function $\mu_i(k)$ is then obtained by integrating $\mu_i'(k)$, which is also a polynomial function.

It is evident that $\mu_i(k)$ satisfies:

$$\mu_i(k_0) = 0 \quad \text{and} \quad \mu_i(k_n) = 1.$$

Finally, conditioned on a sampled noise ϵ, the modified soft K-Flow flow at time $t \in [0, k_n - k_0]$ is expressed as:

$$\Psi_{k_0+t} = \mathcal{F}^{-1}\left(\sum_i \psi_i(k_x, k_y, k_z) \cdot \mu_i(k_0 + t) \cdot \mathcal{F}\{\phi\}(k_x, k_y, k_z) + \sum_i \psi_i(k_x, k_y, k_z) \cdot (1 - \mu_i(k_0 + t)) \cdot \epsilon \right). \tag{20}$$

Through the application of this formula and a family of soft bump functions $\{\mu_i\}$, we can also implement algorithm 1. In comparison to the hard bump functions, a K-Flow constructed with soft bump functions assigns varying weights to each scale according to the scaling step k. Unlike hard bump functions which strictly set other scales to zero for each stage of k, soft bump functions provide a more gradual transition, allowing for multiple frequencies to flow concurrently, and the relative weights are determined by the current scaling parameter k.

Comments on Haar and Meyer wavelet K-amplitude One type of wavelet that offers both frequency and spatial localization is the Meyer wavelet. The Meyer wavelet is originally defined in the Fourier frequency domain, making it ideal for smooth frequency transitions.

The 1D Meyer wavelet $\psi(t)$ and its scaling function $\phi(t)$ are defined via their Fourier transforms, $\hat{\psi}(\omega)$ and $\hat{\phi}(\omega)$, respectively. The Meyer wavelet is constructed to ensure that the wavelet transform will partition the frequency domain into octave bands.

The Fourier transform of the scaling function $\hat{\phi}(\omega)$ is defined as:

$$\hat{\phi}(\omega) = \begin{cases} 1 & \text{if } |\omega| \leq \frac{2\pi}{3}, \\ \cos\left(\frac{\pi}{2}\nu\left(\frac{3|\omega|}{2\pi} - 1\right)\right) & \text{if } \frac{2\pi}{3} < |\omega| \leq \frac{4\pi}{3}, \\ 0 & \text{if } |\omega| > \frac{4\pi}{3}, \end{cases} \tag{21}$$

where $\nu(t)$ is a smooth function defined as:

$$\nu(t) = \begin{cases} 0 & \text{if } t \leq 0, \\ t & \text{if } 0 < t < 1, \\ 1 & \text{if } t \geq 1. \end{cases} \tag{22}$$

The Fourier transform of the Meyer wavelet $\hat{\psi}(\omega)$ is then defined as:

$$\hat{\psi}(\omega) = \begin{cases} \sin\left(\frac{\pi}{2}\nu\left(\frac{3|\omega|}{2\pi} - 1\right)\right) & \text{if } \frac{2\pi}{3} < |\omega| \leq \frac{4\pi}{3}, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

In other words, Meyer transformation can be seen as the Fourier transform with a spatial cutoff window. Note that the scaling function and the wavelet function play different roles, where the low-frequency content of data are obtained by convolving the signal with the scaling function.

In the ablation section, we will employ a specific discretization of the Meyer wavelet to generate our data. Additionally, we will explore the Haar wavelet method, which is implemented solely through spatial convolution kernels and scaling operations. The Haar wavelet, being the simplest form of wavelet, is particularly interesting because it uses piecewise constant functions to capture local features at varying scales, providing a contrast to the smoother Meyer wavelet.

C.3 DWT TRANSFORM AS A K-FLOW DECOMPOSITION

The Discrete Wavelet Transform (DWT) [2] is utilized to decompose a signal at multiple scales, capturing both time and frequency characteristics. It involves scaling and translating wavelets.

The DWT decomposes the input signal into approximation and detail coefficients:

- Given a discrete signal $x[n]$ (expressed by a finite-dimensional vector), use the scaling function $\phi(t)$ and wavelet function $\psi(t)$ to generate coefficients:

$$c_k[j] = \sum_n x[n] \cdot \phi_{k,j}[n]$$
$$d_k[j] = \sum_n x[n] \cdot \psi_{k,j}[n]$$

Here, $c_k[j]$ are the approximation coefficients at scale k, and $d_k[j]$ are the detail coefficients at scale k. Comparing with our definition of K-Flow decomposition, k is just a discrete scaling parameter.

The inverse transform then reconstructs the original signal from the coefficients:

$$x[n] = \sum_k c_j[k]\phi_{j,k}[n] + \sum_k d_j[k]\psi_{j,k}[n]$$

Recursive Relationship between different Scales (k) Different levels of decomposition are recursively related:

1. $k = 1$: A single level decomposition results in approximation coefficients c_1 and detail coefficients d_1;
2. $k = 2$: A two-level decomposition first produces coefficients c_1 and d_1. Then, the approximation coefficients c_1 are further decomposed into a second level of approximation coefficients c_2 and detail coefficients d_2.

For $k = 2$, the decomposition looks like: $x[n] \rightarrow (c_2, d_2, d_1)$, where d_1 represents the high-frequency components (level 1 detail coefficients) and c_1 is the low-frequency component (level 1 approximation coefficients). Further decomposing c_1 yields c_2 (level 2 approximation coefficients) and d_2 (level 2 detail coefficients). This recursive relationship illustrates why we can effectively take a finite maximum scaling k_{max} and still own an inverse transform.

Experiment implementation In this paper's experiments, especially the Wavelet version of K-Flow flow, we take the k_{max} to be one or two. One means decomposing the data into two scales, and two means decomposing the data into three scales.

C.4 IMPLEMENTATION DETAILS

Model design Despite this model-agnostic nature, the unique K-amplitdue localization property of Equation (9) offers an opportunity to design more efficient models. For instance, consider points that lie outside the support of function $\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\in[\lfloor k\rfloor,\lfloor k\rfloor+1)}$. In these regions, their derivative remains zero, indicating that they do not contribute to the optimization process for the corresponding scaling band. This selective activation allows us to focus computational efforts solely on the values within the support of the indicator function, $\mathbb{I}_{\sqrt{k_x^2+k_y^2+k_z^2}\in[\lfloor k\rfloor,\lfloor k\rfloor+1)}$. By doing so, the values outside this region can be treated as static conditions. , providing a fixed context.

Training hyper-parameters In our experiments, we use the pretrained VAE from Stable Diffusion [50]. The VAE encoder has a downsampling factor of 8 given an RGB pixel-based image $\mathbf{x} \in \mathbb{R}^{h\times w\times 3}$, $\mathbf{z} = \mathcal{E}(\mathbf{x})$ has shape $\frac{h}{8} \times \frac{w}{8} \times 4$. All experiments are operated in the fixed latent space.

In Table 4, we provide training hyperparameters for the image generation tasks on the two datasets. For implementing training algorithm Algorithm 1, the bump function is provided in eq. (18). For the classifier-free sampling on the conditional generation task, the cfg-scale is set to be 1.5.

Table 4: Hyper-parameters of DiT network.

	CelebA 256	**IMNET**
Model	DiT-L/2 [46]	DiT-L/2 [46]
lr	2e-4	1e-4
AdamW optimizer (β_1 & β_2)	0.9, 0.999	0.9, 0.999
Batch size	32	240
# of epochs	500	900
# of GPUs	2	16

D ALGORITHMS

In this section, we list three key algorithms.

Algorithm 1 Training of K-Flow.

```
Require: scaling parameter $k$ with maximum $k_{max}$, K-Flow transform $\mathcal{F}$, inverse transform $\mathcal{F}^{-1}$, noise distribution $p$, target distribution $q$
Normalize $k$ to be in $[0, 1]$: $k \leftarrow k/k_{max}$
Initialize parameters $\theta$ of $v_k$
while not converged do
    sample scaling $k \sim \mathcal{U}(0, 1)$
    sample training example $\phi \sim q$, sample noise $\epsilon \sim p$
    Calculate current flow position $\Psi_k$ according to K-Flow transform $\mathcal{F}$, $\mathcal{F}^{-1}$ and Equation (8)
    Calculate the conditional vector field $\dot{\Psi}_k$ according to $\mathcal{F}$, $\mathcal{F}^{-1}$ and Equation (9)
    Calculate the objective $\ell(\theta) = \|v_k(\Psi_k; \theta) - \dot{\Psi}_k\|_g^2$, following Equation (10)
    $\theta$ = optimizer_step($\ell(\theta)$)
end while
```

Algorithm 2 Scaling-Controllable Generation of K-Flow.

```
Require: Scaling parameter $k$, $K$-amplitude transform $\mathcal{F}$, inverse transform $\mathcal{F}^{-1}$, noise distribution $p$ in the $K$-amplitude space, target distribution $q$
Initialize pre-trained $v_k(\theta)$
Sample one high-scaling noise $\epsilon_{\text{high}} \sim p$, sample two independent low-scaling noise $\epsilon_{\text{low}} \sim p$, $\tilde{\epsilon}_{\text{low}} \sim p$
$\Psi_0 = \mathcal{F}^{-1}\{\epsilon_{\text{low}}, \epsilon_{\text{high}}\}$
$\tilde{\Psi}_0 = \mathcal{F}^{-1}\{\tilde{\epsilon}_{\text{low}}, \epsilon_{\text{high}}\}$
for $k \in [0, 1]$ do
    $\Psi_k \leftarrow$ ODEstep($v_k(\cdot, \theta)$, $\Psi_0$)
    $\tilde{\Psi}_k \leftarrow$ ODEstep($v_k(\cdot, \theta)$, $\tilde{\Psi}_0$)
end for

return $\Psi_1$, $\tilde{\Psi}_1$
```

Algorithm 3 Class-conditional Generation of K-Flow with dropping.

```
Require: Pre-trained $v_k(\theta)$, conditioning class $c$, dropping time $\tau$, noise distribution $p$, guidance parameter $\omega$
$\Psi_0 \sim p$
for $k \in [0, \tau]$ do
    $\tilde{v}_k(\cdot) \leftarrow (1 - \omega)v_k^\theta(\cdot, \theta) + \omega u_k(\cdot, c, \theta)$  {guided velocity}
    $\Psi_\tau \leftarrow$ ODEstep($\tilde{v}_k(\cdot)$, $\Psi_0$)
end for
for $k \in [\tau, 1]$ do
    $\Psi_1 \leftarrow$ ODEstep($v_k(\cdot, \theta)$, $\Psi_\tau$)
end for

return $\Psi_1$
```

E MORE RESULTS

E.1 MORE RESULTS ON UNCONDITIONAL GENERATION

We provide more results on the class-conditional generation in Figure 9.

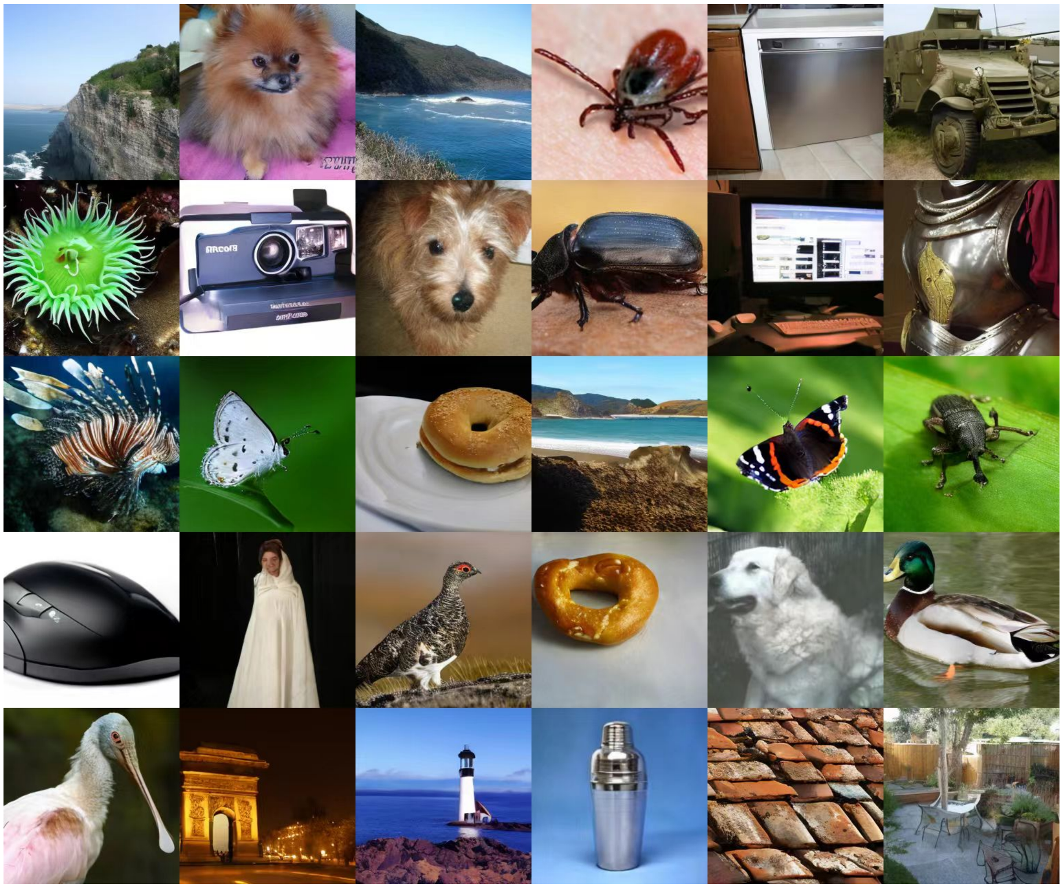


Figure 9: Non-curated samples of our reversing scaling variant on ImageNet (cfg = 1.5).

E.2 ABLATION STUDIES ON CONTROLLABLE CLASS-CONDITIONAL GENERATION

In Section 4, we provide brief results on the controllable class-conditional generation over ImageNet. Here in Figure 10, we would like to give a more qualitative comparison between our model K-Flow and LFM.

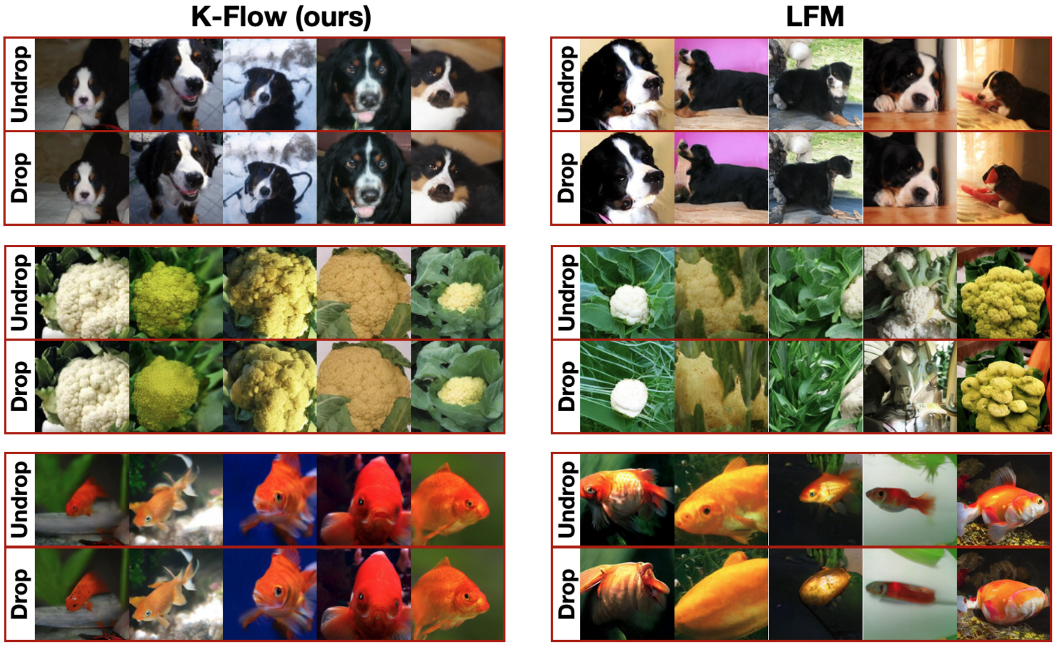
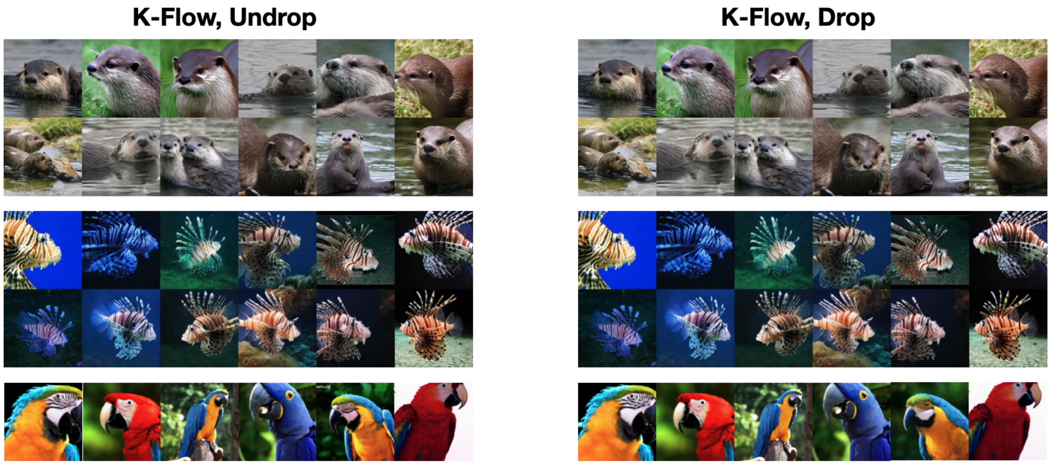


Figure 10: Results of controllable class-conditional generation, with hyperparameter cfg=1.5.



K-Flow, Undrop

K-Flow, Drop

Figure 11: Classifier-free guidance sampling of our Fourier-based K-Flow with a hyperparameter setting of cfg $= 3$. In the right columns, the class condition is omitted for the last 50% of the scaling steps during inference, using the same initial noise. It can be observed that as the cfg value increases and the duration of omitting the class condition decreases, the generated results appear nearly identical.

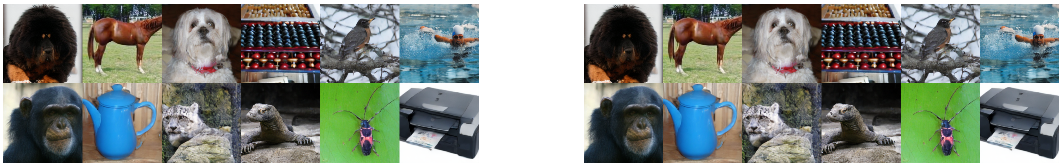


Figure 12: Classifier-free guidance sampling of our wavelet-based K-Flow with a hyperparameter setting of cfg $= 2$. In the right columns, the class condition is drooped for the last 70% of the scaling steps during inference, using the same initial noise. It can be observed that after dropping, K-Flow still preserves the high-scaling contents.

E.3 ABLATION STUDY ON WAVELET BASE

From Table 5, we tested two additional wavelet base, the discrete Haar basis [21] and the discrete Myer basis [43] as a supplement of the Daubechies wavelet (db6) used in the main text. All three wavelets demonstrated comparable performance in terms of both the FID and Recall metrics.

E.4 ABLATION STUDY ON SCALING PARTITIONS

Although the quality of face generation appears similar to the naked eye, the model with three K-amplitude bands (the last row of Table 5) performed worse in terms of FID and Recall metrics. We provide the generated samples for qualitative evaluation in Figure 13. **Reversing the K-amplitude scaling** In Table 6, we also tested a counterintuitive scaling order: from high to low. This means generating high-frequency details first and then filling in the low-frequency components during the generation process. We find that the model can still produce images normally (Figure 9), with a better diversity (Recall) but lower quality (FID) compared to the low-to-high scaling approach.

E.5 PIPELINE OF SCALING-CONTROLLABLE GENERATION

We first demonstrate the pipeline of scaling-controllable generation in Figure 14. In contrast to Figure 3b, here K-Flow flows from high-scaling to low-scaling, thus we can keep the noise corresponding to the low-scaling fixed while changing the high-scaling part.

Table 5: CelebA-HQ 256

Model	FID↓	Recall↑
CelebA-HQ 256		
K-Flow, Meyer-DiT L/2	**5.01**	**0.47**
K-Flow, Haar-DiT L/2	5.01	0.46
K-Flow, Db-DiT L/2 (three scales)	5.77	0.42

Table 6: Conditional ImageNet (256)

Model	FID↓	KSR↓	Recall↑
K-Flow, Wave (reverse)	23.06	-	0.58
+ cfg=1.5	5.1	-	0.46
K-Flow, Wave-DiT L/2 (Ours)	17.8	-	0.56
+ cfg=1.5	4.49	-	0.44
LFM (DiT L/2)	14.0	-	0.56
+ cfg=1.5	2.78	-	0.45

Figure 13: Daubechies wavelet K-amplitude with more components trained on CelebA-256

Remarks on the steerable experiment. Although the overall results are generally optimistic, some unexpected changes have been observed in the high-scaling parts. This may be attributed to two factors:

1. The compressed latent space may mix high and low content present in the original pixel space.
2. The loss Equation (10) may not be perfectly optimized, meaning that K-Flow localized vector field might not be perfectly confined to the low-scaling part. The second factor might be mitigated by training on larger datasets. Furthermore, by training a reversed K-Flow flow (from high to low), we observe that fixing the low-scaling noise enables unsupervised editing of detailed high-scaling content.

In Figure 4b, we've tested the wavelet-based K-Flow and observed similar results with the Fourier-based K-Flow. However, for PCA, we couldn't identify obvious semantic edits that are interpretable to human eyes (see Figure 15). This might be because PCA scaling doesn't align well with multi-resolution inductive biases.

This insight further supports our model's capacity to decompose the generative process into distinct frequency bands, where specific frequency bands can be independently controlled. This separation aids in achieving more detailed and deliberate modifications to generated data, adding a layer of precision and flexibility to the generative framework.

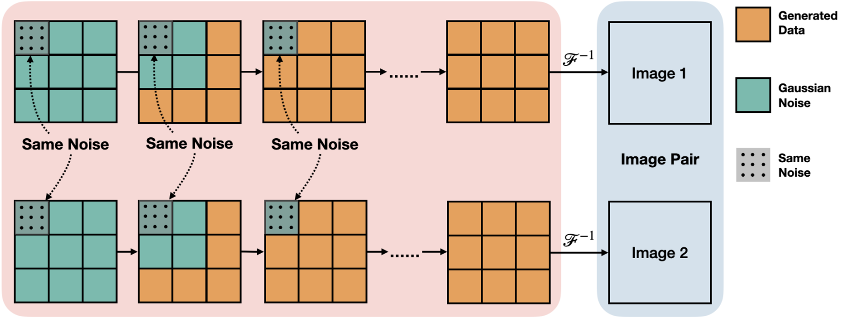


Figure 14: Pipeline of scaling-controllable generation (high scaling).

Figure 15: PCA editing by Algorithm 2.

E.6 CLASS-CONDITIONAL FID

We propose using the class-aware FID metric, defined as follows:

$$\text{FID}_{\text{class-conditional}} = \mathbb{E}_{c\sim p(c)}\left[\text{FID}(c)\right] \tag{24}$$

where for each class c, the FID is calculated by:

$$\text{FID}(c) := \text{FID}(X_r^c, X_g^c) = \|\mu_r^c - \mu_g^c\|^2 + \text{Tr}(\Sigma_r^c + \Sigma_g^c - 2(\Sigma_r^c\Sigma_g^c)^{1/2}). \tag{25}$$

Here, X_r^c and X_g^c denote the real and generated data subsets for class c, respectively. Based on $\text{FID}(c)$, the Class-Dropping-Ratio (CDR) is defined by

$$\text{CDR} := \mathbb{E}_{c\sim p(c)}\left[\frac{\text{FID}_{\text{bef}}(c)}{\text{FID}_{\text{aft}}(c)}\right],$$

where FID_{bef} denotes the FID calculated for the flow model carried with the class condition for the whole process, and FID_{aft} denotes the FID calculated for the flow model carried with the class condition for only a subprocess (we keep the initial 30% of the inference time for the experiment). In practice, instead of computing the expectation over the entire class distribution $p(c)$, we randomly select 5 classes out of the total 1000 classes for evaluation.