Master of Science in Industrial and Applied Mathematics
Université Grenoble Alpes

Hierarchical Generative Modeling via Wavelet Decomposition and Flow Matching

Maxime Attwood

Defence : June 2025

Research project performed in the Robotlearn team at INRIA Grenoble
February 2025 – June 2025

Under the supervision of:
Samir Sadok
Xavier Alameda-Pineda

Abstract

High-resolution image synthesis is challenging due to high computational cost and the difficulty of capturing both global structure and fine details. In this work, we introduce Wavelet-FM, a hierarchical generative model that addresses these issues by combining Flow Matching (an efficient continuous-flow generation approach) with wavelet-based multi-resolution decomposition. Wavelet-FM first generates a low-frequency coarse image using a Flow Matching model. It then progressively adds high-frequency details via a second model conditioned on the coarse output, recombining them through an inverse wavelet transform. This two-stage design significantly improves generation efficiency and image quality. In experiments on standard image datasets (such as CIFAR-10 and CelebA), our method produces images with great fidelity using fewer sampling steps than other Flow Matching methods with a competitive FID of 7.95 on CIFAR-10 with only 35 function evaluations. Notably, adding learned high-frequency components to coarse images dramatically enhanced realism—reducing the Fréchet Inception Distance. The modular architecture also allows the fine-detail generator to generalize across different datasets, indicating a decoupling of structure and texture information. Overall, the proposed hierarchical framework yields fast, high-quality image generation and provides explicit control over image resolution and detail, with promising applications in scalable image synthesis and editing.

Résumé

La synthèse d'images haute résolution reste un défi en raison de son coût computationnel élevé et de la difficulté à capturer à la fois la structure globale et les détails fins. Dans ce travail, nous présentons *Wavelet-FM*, un modèle génératif hiérarchique qui répond à ces enjeux en combinant *Flow Matching*, une méthode efficace de génération par flux continu, avec une décomposition multi-résolution basée sur les ondelettes. Notre approche génère d'abord une image de basse fréquence (LL) à l'aide d'un modèle Flow Matching, puis ajoute progressivement les détails haute fréquence (HF) via un second modèle conditionné sur l'image initiale. L'image finale est reconstruite par transformée en ondelettes inverse, permettant une génération modulaire, récursive et adaptée à la résolution cible.

Les expériences menées sur des ensembles de données standards (CIFAR-10, CelebA) montrent que notre méthode produit des images de grande fidélité tout en nécessitant moins d'étapes d'échantillonnage que les approches Flow Matching classiques, seulement 35 évaluations pour un FID de 7.95. L'ajout de composantes HF apprises améliore nettement le réalisme. L'architecture modulaire permet également au générateur de détails fins de généraliser à travers différents jeux de données, révélant une séparation explicite entre la structure et la texture. Ce cadre hiérarchique ouvre ainsi des perspectives prometteuses pour la génération d'images rapide, contrôlable et extensible à des résolutions variées.

Acknowledgements

I would like to express my sincere gratitude to my internship supervisors, Xavier Alameda-Pineda and Samir Sadok. I thank Xavier for offering me the opportunity to join the Robotlearn team and for his high-level guidance throughout the project. I am especially grateful to Samir for his daily support, continuous availability, and insightful feedback, which were crucial throughout the development and implementation of this work.

I am also grateful to the Robotlearn team at Inria Grenoble for the warm welcome, stimulating discussions, and supportive environment that made this internship both productive and enjoyable.

Contents

1 Introduction

Context and Motivation

Generative modeling has witnessed remarkable progress over the last decade, driven by the emergence of diverse paradigms such as **Generative Adversarial Networks** (GANs, Goodfellow et al. (2014)), **Variational Autoencoders** (VAEs, Kingma and Welling (2022)), and, more recently, **diffusion-based models**. GANs introduced an adversarial training framework that produces high-fidelity samples but often suffers from mode collapse. VAEs provide a principled likelihood-based approach but tend to yield blurrier outputs. Diffusion models (Sohl-Dickstein et al. (2015), Ho et al. (2020)) have set new state-of-the-art in image synthesis by iteratively denoising data, yet they often require hundreds of sampling steps to generate a single image.

These advances have unlocked a wide range of applications, from photorealistic image synthesis and artistic content creation to data augmentation for downstream tasks and even scientific imaging in fields such as medical diagnostics and remote sensing. However, the growing demand for higher-resolution outputs and real-time generation exposes critical limitations in existing methods, particularly regarding computational cost and the ability to capture both global structure and fine-grained details.

One key challenge lies in **multi-scale image generation**: naively generating high-resolution images scales poorly with resolution, leading to prohibitive memory and time requirements. While diffusion models can, in principle, generate arbitrary resolutions, their heavy sampling requirements exacerbate this issue. Recent work has proposed various acceleration techniques, yet the literature continues to highlight the need for more efficient sampling without degrading sample quality.

A natural solution is to adopt a multi-resolution or hierarchical approach: instead of generating the full image at once, one can produce a **coarse representation** and then refine it progressively. By conditioning the generation of **finer details** on previously produced coarser features, we can maintain coherence across scales and reduce the overall computational burden. Additionally, a patch-wise generation strategy—whereby the image is generated patch by patch and each patch is conditioned on its neighboring regions—can further decrease memory requirements and ensure local consistency without processing the entire high-resolution image in one go.

In parallel, **Flow Matching** (Lipman et al., 2023) has emerged as a promising alternative to diffusion-based sampling, formulating generation as the integration of a time-dependent vector field that transports a simple prior to the data distribution. Flow Matching achieves comparable sample quality with significantly fewer function evaluations, addressing the inefficiency of standard diffusion pipelines. Moreover, hierarchical schemes offer explicit control over generation at different frequency bands, which is desirable in applications requiring interpretability or targeted editing.

To enable principled multi-scale decomposition, we leverage **wavelet and multi-wavelet transforms**, classical tools in signal processing and image compression that decompose signals into frequency-localized components. By combining Flow Matching with a wavelet-based multi-resolution analysis, our pipeline can separately model low- and high-frequency content, yielding faster sampling and improved fidelity across scales.

Despite these advances, no existing framework simultaneously provides fast sampling and multi-scale generation using Flow Matching. In this report, we address this gap by introducing **Wavelet-FM**, a hierarchical generative modeling pipeline that integrates Flow Matching with wavelet-based multi-resolution decomposition.

Problem Statement

While recent works have addressed either sampling inefficiency or resolution scalability in generative modeling, few approaches attempt to unify both within a coherent and interpretable framework. In particular, there remains a lack of principled models that can gen-

erate high-resolution images efficiently while explicitly capturing information at multiple frequency scales.

Here are the main challenges we aim to resolve :

- **Sampling inefficiency**: Traditional diffusion models require hundreds of iterative steps to sample a single image, which incurs significant computational cost. We explore Flow Matching as an alternative approach, which models continuous flows and achieves comparable quality with fewer function evaluations.
- **Multi-scale consistency**: Naively generating images at high resolution is inefficient and can lead to poor global structure. Our objective was to decompose image generation into semantically meaningful frequency bands using wavelet transforms, separating low-frequency structure from high-frequency detail.
- **Modular architecture**: We sought to build a two-stage generative system where a first model generates low-frequency (LL) components and a second model generates high-frequency (HF) wavelet bands conditioned on the LL image and resolution. This separation improves modularity, interpretability, and training scalability.
- **Generalization across scales and domains**: We aimed for the HF generator to generalize across multiple resolutions and datasets, enabling a recursive upsampling process and cross-domain reuse.
- **Efficiency and token optimization**: Inspired by recent K-Flow work, we additionally explored how selective token feeding and progressive scale-wise inference can reduce token evaluations without sacrificing quality.

We aim to address these issues by designing a principled hierarchical generation scheme that:

- Separates low- and high-frequency components through wavelet-based decomposition.
- Uses Flow Matching to efficiently learn continuous flows for both coarse and fine components.
- Enables scale-wise conditioning to maintain coherence and reduce memory costs.

Objectives and Contributions

The objective of this work is to design and evaluate a hierarchical generative model that combines Flow Matching with wavelet-based decomposition to enable efficient, high-quality image synthesis across multiple resolutions. Our approach, called Wavelet-FM, aims to address both the computational inefficiencies of diffusion models and the lack of explicit multi-scale structure in standard generative frameworks.

The main contributions of this work are as follows:

- We propose a novel architecture, **Wavelet-FM**, that integrates a two-stage generation process: first generating a coarse low-frequency image using Flow Matching, then generating high-frequency components conditioned on it.
- We leverage wavelet decomposition to split images into interpretable low- and high-frequency bands, enabling separate modeling and reconstruction via inverse transforms.
- We introduce a conditional U-Net based on a modified version of the original U-Net (Ronneberger et al., 2015) that generates high-frequency wavelet bands conditioned on both the low-resolution image and the target resolution level.

- We conduct extensive experiments on CelebA (Liu et al., 2015) and CIFAR-10 (Krizhevsky and Hinton, 2009) , comparing performance with baseline Flow Matching models and analyzing generation quality across scales.
- In parallel, we explored an extension based on K-Flow (Du et al., 2025) , a recent coarse-to-fine generation approach using DiT (Peebles and Xie, 2023) models and amplitude-based interpolation. While prior work feeds full-resolution patchified tokens including noisy regions, we investigate a variant where only the informative regions to be filled are provided to the model, potentially reducing the token count and improving efficiency. This line of work remains ongoing beyond the defense period.

Overall, the objective was to investigate whether combining wavelet decomposition with Flow Matching could yield an efficient, hierarchical, and interpretable generative modeling pipeline.

Report Organization

This report is structured as follows. Section 2 provides a review of the state of the art in generative modeling, covering GANs, VAEs, diffusion models, and flow-based approaches such as Flow Matching and K-Flow. It also introduces architectural components like U-Nets and DiTs, and summarizes commonly used evaluation metrics. Section 3 presents empirical insights into Flow Matching, including interpolation trajectories, PCA analysis, class conditioning, and performance on CIFAR-10. Section 4 introduces our main contributions: a modular wavelet-based generative architecture. We first describe and evaluate a single-scale pipeline, then extend it to a multi-scale generator. This section also includes a theoretical analysis of token efficiency in the K-Flow framework. Finally, Section 5 concludes the report with a discussion of contributions, limitations, and future research directions.

2 State-Of-The-Art

2.1 Overview of Generative Models

Generative modeling has evolved through multiple paradigms, including Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs). Although VAEs offer a probabilistic framework and GANs have demonstrated impressive sample quality, both approaches face challenges, such as blurry reconstructions (VAEs) or unstable training and lack of likelihoods (GANs).

In recent years, score-based methods, such as diffusion models, and their efficient counterparts like Flow Matching have emerged as promising alternatives. This chapter focuses on these newer methods that underpin the design of our proposed architecture.

2.2 Diffusion Models

Diffusion models are a class of generative models that generate data by gradually transforming noise into meaningful samples. They define a *forward process*, which adds noise to data, and a *reverse process*, which learns to denoise and recover the original distribution.

Forward Process (Noise Injection)

Given a data sample x_0, the forward process progressively corrupts it with Gaussian noise over T time steps:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I) \tag{1}$$

where α_t is a noise schedule that controls the variance of added noise. As t increases, the data distribution converges to an isotropic Gaussian distribution.

Figure 1: Visualization of the forward process in diffusion models. The image gradually transitions from a clean input (left) to pure Gaussian noise (right) as more noise is applied.

Reverse Process (Denoising)

The goal of diffusion models is to approximate the reverse process, which removes noise step by step:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2}$$

where $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are neural networks (typically U-Nets) trained to predict denoised samples.

Training via Score Matching

Rather than explicitly modeling likelihoods, diffusion models use *score-based training* by optimizing a noise-prediction objective:

$$L = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right] \tag{3}$$

where $\epsilon_\theta(x_t, t)$ is the model's noise prediction network, and $\epsilon \sim \mathcal{N}(0, I)$ is sampled Gaussian noise.

Sampling and Limitations

Despite their success, diffusion models suffer from significant drawbacks. The reliance on a discrete denoising process restricts the sampling trajectories to a confined probability space, resulting in prohibitively long training times and inefficient sampling[1].

This limitation has been widely discussed in recent works, with various methods proposed to accelerate the reverse process (Song et al., 2022; Zhang and Chen, 2023). Lipman et al. (2023) highlight that the reliance on simple diffusion processes contributes to these inefficiencies, requiring specialized methods to improve sampling speed.

2.3 Flow Matching

2.3.1 General Concepts

Flow Matching (FM) is a generative modeling approach that constructs a smooth probability path $(p_t)_{0 \leq t \leq 1}$, which continuously transforms an initial distribution p_0 into the target distribution p_1. This is achieved by learning a time-dependent velocity field $u_t(x)$, modeled by a neural network, which dictates how samples should evolve through time.

Flow Matching is derived from *Continuous Normalizing Flows* (CNFs), a class of generative models that parameterize probability distributions using a continuous transformation governed by an Ordinary Differential Equation (ODE):

$$\frac{dx_t}{dt} = u_t(x_t), \tag{4}$$

where $u_t(x)$ is a time-dependent velocity field. CNFs allow for arbitrary smooth probability paths, including diffusion-based probability paths (Onken et al., 2021).

Using this formulation, Flow Matching describes the transformation from the initial distribution with the **flow function** $\psi_t(x)$, which follows the ODE:

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)), \quad \psi_0(x) = x. \tag{5}$$

Since the velocity field $u_t(x)$ fully dictates the evolution of $\psi_t(x)$, knowing one allows the complete reconstruction of the other. In other words, solving for $\psi_t(x)$ given $u_t(x)$ is done by integrating:

$$\psi_t(x) = x + \int_0^t u_s(\psi_s(x))ds. \tag{6}$$

Thus, the velocity field and the flow function are equivalent representations of the same underlying transformation.

The **velocity field** $u_t(x)$ is trained to ensure that solving this ODE transports samples from p_0 to p_1. Once trained, new samples are generated by:

- Sampling $X_0 \sim p_0$.
- Using the learnt velocity field u_t^θ to estimate $u_t(\psi_t(x))$
- Solving the ODE until $t = 1$, to obtain ψ_1, usually through discrete schemes (*i.e.* Euler's method)
- Finally, obtaining $X_1 = \psi_1(X_0)$

[1] In the context of generative models, "sampling" refers to the process of generating a new data point (e.g., an image) from the learned data distribution, typically by transforming a random noise vector using the trained model.

The evolution of a sample $X_0 \sim p_0$ under the flow function $\psi_t(x)$ defines a smooth **probability path** $(p_t)_{0 \leq t \leq 1}$, satisfying:

$$X_t := \psi_t(X_0) \sim p_t, \quad \text{for } X_0 \sim p_0. \tag{7}$$

At each time t, the transformed variable X_t follows the intermediate distribution p_t:

$$X_t \sim p_t. \tag{8}$$

This probability path continuously transports the source distribution p_0 to the target distribution p_1 and the objective of Flow Matching is to optimize u_t^θ, such that its flow ψ_t correctly models such a probability path.

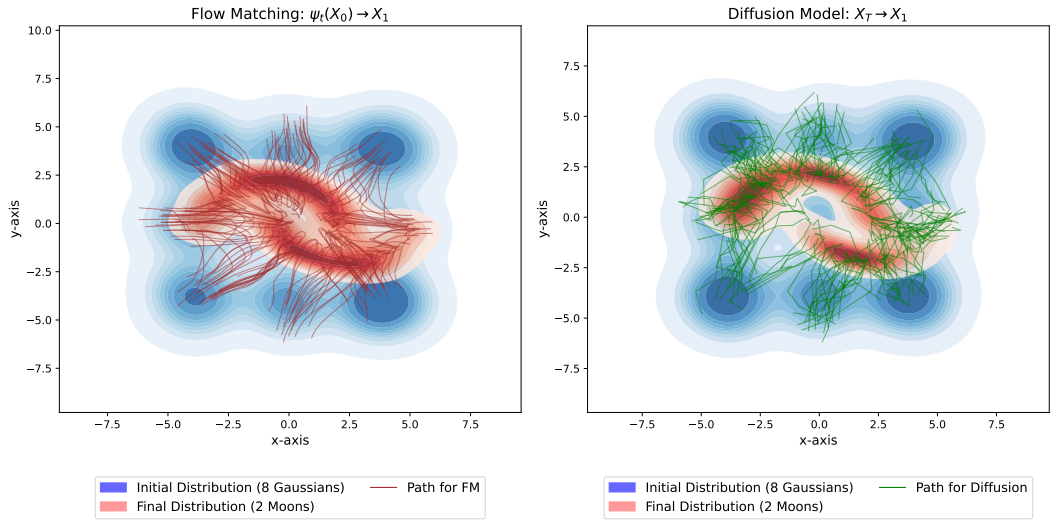

Figure 2: Comparison of Flow Matching (left) and Diffusion (right). The blue KDE represents the **initial distribution** (8 Gaussians), and the red KDE represents the **final distribution** (2 Moons). The brown curves show **transport paths for Flow Matching**, while the green curves represent **diffusion trajectories**. The transport paths were plotted using 30 function evaluations (NFE) for the Diffusion model and a 30-step Euler discretization for the Flow Matching model.

Figure 2 highlights the differences between Flow Matching (FM) and Diffusion-based models in mapping an initial distribution (8 Gaussians) to a target distribution (2 Moons). A key distinction is that FM produces smooth, continuous transport paths (brown curves), whereas diffusion follows noisy trajectories (green curves). Additionally, FM required only 20,000 training epochs, while the diffusion model trained for 50,000 epochs, reflecting the increased complexity of learning a stochastic denoising process. However, while FM paths appear structured, they are not necessarily optimal, suggesting potential improvements using Optimal Transport methods, which will be explored in a later section.

2.3.2 Designing the Path

In practice, while we know the initial data, which follows a known distribution, and the final data, of which we have samples, we cannot directly draw samples from the intermediate distribution p_t for $0 < t < 1$.

There exist multiple ways to define a path, two of them have been explored in initial paper on FM (Lipman et al., 2023).

One way to construct a meaningful probability path, is to define the **conditional optimal-transport (OT) path**, also known as the **linear path**. This path provides

a structured interpolation between the source distribution p_0 and the target distribution p_1, ensuring a smooth transition:

$$p_t(x) = \int p_{t|1}(x|x_1)q(x_1)dx_1, \quad \text{where } p_{t|1}(x|x_1) = \mathcal{N}(x|tx_1, (1-t)^2 I). \tag{9}$$

This formulation describes a Gaussian distribution centered at tx_1 with variance scaling as $(1-t)^2$, ensuring a gradual transition from p_0 to p_1.

Using this probability path, we can generate intermediate samples by linearly interpolating between the source and target distributions:

$$X_t = tX_1 + (1-t)X_0, \quad \text{where } X_0 \sim p_0, \quad X_1 \sim p_1. \tag{10}$$

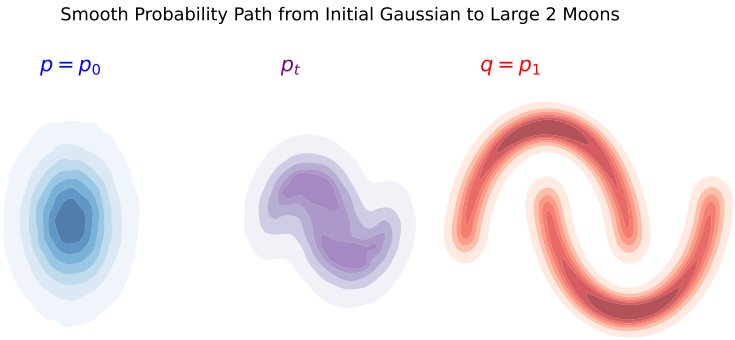

Figure 3: Visualization of the probability path from an initial Gaussian (left) to a large 2-Moons distribution (right). The probability path p_t smoothly interpolates between p_0 and p_1 and is the evolution of the probability densities over time.

2.3.3 Objective Function

As explained earlier, the goal of Flow Matching is to optimize u_t^θ, to correctly model the probability path p_t defined in the last section.

This is achieved by regressing u_t^θ to match the true velocity field u_t, which generates p_t. The **Flow Matching loss** is thus formulated as:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,X_t} \left\| u_t^\theta(X_t) - u_t(X_t) \right\|^2, \tag{11}$$

where $t \sim \mathcal{U}[0,1]$ and $X_t \sim p_t$.

In practice, directly optimizing this objective is impossible. Instead, we simplify the loss by conditioning on a single target example $X_1 = x_1$, randomly drawn from the training data.

The Flow Matching loss can be efficiently minimized by training a neural network to approximate $u_t(x|x_1)$ given samples X_t. This gives us the **Conditional Flow Matching loss**, formulated as:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_t,X_1} \left\| u_t^\theta(X_t) - u_t(X_t|X_1) \right\|^2, \tag{12}$$

where $t \sim \mathcal{U}[0,1]$, $X_0 \sim p$, and $X_1 \sim q$.

Unlike the FM objective, the CFM objective allows us to easily sample unbiased estimates as long as we can efficiently sample from $p_t(x|x_1)$ and compute $u_t(x|x_1)$, both of which can be done easily.

In particular, the gradient of this **Conditional Flow Matching loss** is equal to the gradient of the original **Flow Matching loss**, ensuring that optimizing either objective leads to the same learned velocity field, while also providing an easier-to-calculate formula:

$$\nabla_\theta \mathcal{L}_{\text{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta). \tag{13}$$

However, we can simplify this loss even further. To do so, we define the conditional random variables :

$$X_{t|1} = tx_1 + (1-t)X_0 \sim p_{t|1}(\cdot|x_1) = \mathcal{N}(\cdot|tx_1, (1-t)^2 I). \tag{14}$$

From this, solving the differential equation $\frac{d}{dt}X_{t|1} = u_t(X_{t|1}|x_1)$ leads to the **conditional velocity field**:

$$u_t(x|x_1) = \frac{x_1 - x}{1-t}. \tag{15}$$

This expression can now be reinjected in expression (12), which gives us :

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_t,X_1}\left\| u_t^\theta(X_t) - \frac{X_1 - X_t}{1-t} \right\|^2, \tag{16}$$

Finally, using expression (10) for X_t, we get :

$$\begin{aligned} \mathcal{L}_{\text{CFM}}^{\text{OT}}(\theta) &= \mathbb{E}_{t,X_0,X_1}\left\| u_t^\theta(X_t) - \frac{X_1 - (tX_1 + (1-t)X_0)}{1-t} \right\|^2 \\ &= \mathbb{E}_{t,X_0,X_1}\left\| u_t^\theta(X_t) - (X_1 - X_0) \right\|^2 \end{aligned} \tag{17}$$

2.3.4 Structured Sampling with Optimal Transport

In the previous part, we saw how to simplify the loss used to train a Flow Matching model. However, X_0 and X_1 were drawn assuming independence. This can lead to suboptimal paths, as seen in Figure 2, where the learned flow does not follow an efficient transport route.

To address this, we aim to sample $q(X_0, X_1)$ in a way that ensures X_0 and X_1 are optimally coupled. Instead of sampling them independently, we want to pair samples such that they follow the minimal-cost transport plan between the two distributions. This results in more structured and natural interpolation paths, leading to a more efficient training process and improved sample quality, as discussed in Lipman et al. (2024)

Finding the Optimal Transport Plan The problem of finding an optimal coupling can be formulated as an *Optimal Transport* (OT) problem, where we seek a transport plan $\Pi^\star(X_0, X_1)$ that minimizes the displacement cost.

Let $\alpha = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $\beta = \frac{1}{n}\sum_{j=1}^n \delta_{z_j}$ be empirical distributions representing the initial and target distributions. To transport mass from α to β, we define a *transport matrix* Π, where Π_{ij} represents the fraction of mass transported from x_i to z_j. The transport plan must satisfy the mass conservation constraints:

$$\sum_j \Pi_{ij} = \frac{1}{n}, \quad \sum_i \Pi_{ij} = \frac{1}{n}. \tag{18}$$

The transport cost is measured using a ground cost matrix C, where each element is given by:

$$C_{ij} = \|x_i - z_j\|^2. \tag{19}$$

Thus, the optimal transport plan is obtained by solving the following minimization problem:

$$\Pi^\star = \arg\min_{\Pi \in U(\alpha,\beta)} \langle \Pi, C \rangle_F, \tag{20}$$

where $U(\alpha, \beta)$ is the set of admissible transport plans satisfying the mass conservation constraints.

Impact on Flow Matching : By using the optimal transport plan $\Pi^\star(X_0, X_1)$, we obtain a structured mapping between samples from $p(X_0)$ and $p(X_1)$. Instead of selecting (X_0, X_1) independently, we sample them according to $\Pi^\star$, ensuring that the interpolation path follows a meaningful transport route.

This leads to **straighter flow trajectories, a more stable training process, and improved efficiency during inference**. The generated samples remain well-structured throughout the evolution from X_0 to X_1, making the learned flow more effective. The difference between regular Flow Matching and OT-FM can be observed in Figure 4.

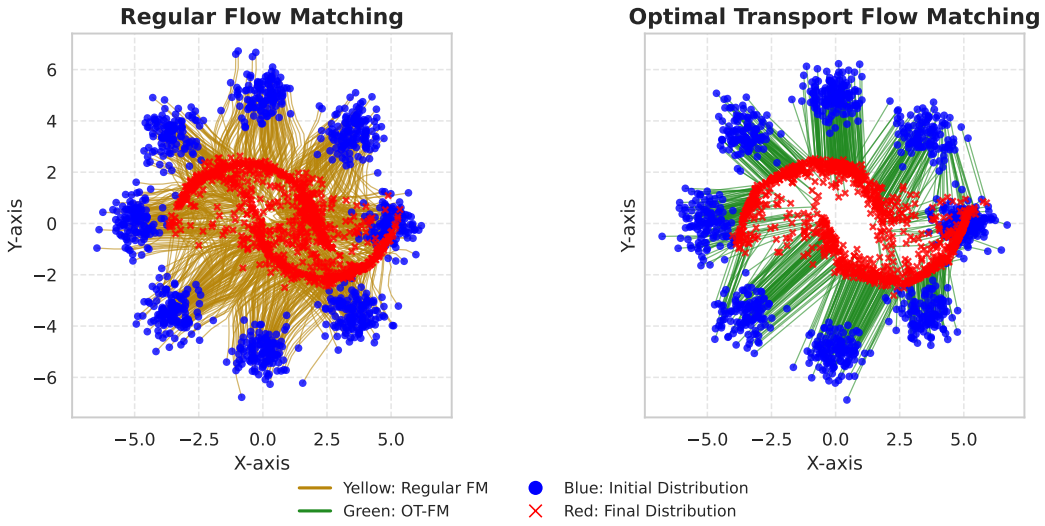

Figure 4: Comparison of Regular Flow Matching (left) and Optimal Transport Flow Matching (right). The blue dots represent the initial distribution (8 Gaussians), and the red dots represent the final distribution (2 Moons). The transport paths are colored brown for Regular FM and green for OT-FM. Both trajectories were computed using Euler, with 2 steps for the OT-FM, and 10 for the regular FM.

2.4 K-Flow

While traditional Flow Matching approaches aim to model a direct continuous flow from noise to data, recent work has explored the possibility of decomposing this path into multiple coarse-to-fine stages. K-Flow, introduced by Du et al. (2025), proposes a framework for progressive generation by introducing a continuous scale parameter $k \in [0, 1]$ that controls the "amplitude" of information revealed at each stage. The model learns a family of flows Φ_k that gradually transport a noisy latent representation into a full-resolution image.

To model the input and output space, K-Flow employs patchified wavelet coefficients and leverages a DiT (Diffusion Transformer) backbone (Peebles and Xie, 2023), allowing

for token-based autoregressive modeling across spatial locations and scales. Unlike classical Flow Matching, which assumes full-resolution conditioning at once, K-Flow defines a dynamic training objective that matches the flow velocity at varying levels of revealed information, enabling coarse-to-fine synthesis in a principled way.

Our own work, developed in parallel and prior to the public release of K-Flow, also puts forward a hierarchical generation pipeline that combines Flow Matching and wavelet decomposition. While K-Flow defines a continuous flow with respect to the information amplitude k, we propose a discrete, two-stage pipeline where low-frequency wavelet components are generated first, and high-frequency bands are then conditioned on them. We also explored a DiT-based variant inspired by K-Flow's patchified formulation. However, unlike the standard K-Flow model—which processes the full image tensor after injecting noise at high-resolution, our approach restricts the input to only the wavelet tokens relevant to the current synthesis scale. This selective token conditioning is designed to reduce redundancy and focus the model's capacity. Thus, our formulation explores a complementary perspective on how coarse-to-fine generation can be structured more efficiently by making it multi-scale as well.

2.5 Architectures in Generative Modeling

Modern generative models rely on expressive neural network architectures to learn mappings from noise to complex image distributions. Two dominant architectural families used in state-of-the-art generative pipelines are convolutional U-Nets and Transformer-based models, particularly Vision Transformers (ViTs). This section briefly reviews their principles and relevance to the models employed in this work.

2.5.1 U-Net Architectures

The U-Net was originally introduced for biomedical image segmentation (Ronneberger et al., 2015) but has since become a standard architecture in generative modeling, especially for diffusion models and flow-based methods. A U-Net is structured as an encoder-decoder architecture with symmetric skip connections:

- The **encoder** progressively downsamples the input by lowering the resolution while increasing the number of channels, capturing hierarchical features through convolutional layers.
- The **decoder** upsamples these representations to the target resolution, combining them with encoder outputs via skip connections to preserve spatial detail.

This design enables multi-scale feature fusion and supports sharp, coherent image generation. U-Nets are widely used in Denoising Diffusion Probabilistic Models (DDPMs) and have also been successfully applied in Flow Matching pipelines. In our work, we employ a conditional U-Net to generate high-frequency wavelet bands, conditioned on a low-resolution base image, as well as the low-frequency wavelet bands themselves.

Because U-Nets are convolutional, they can naturally operate on inputs of varying spatial resolution without modifying their architecture. This flexibility is particularly valuable in our framework, where the model is trained to generate high-frequency wavelet bands at different image scales. Using a single U-Net, we can handle multiple target resolutions efficiently and coherently.

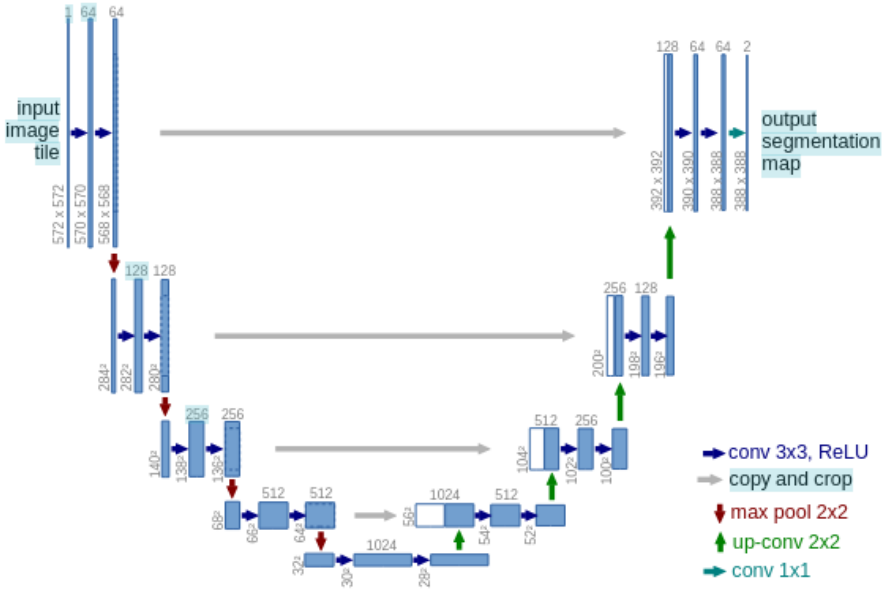

Figure 5: Original U-Net architecture from Ronneberger et al. (2015), showing the symmetric encoder-decoder design with skip connections.

2.5.2 Vision Transformers (ViT) and DiT

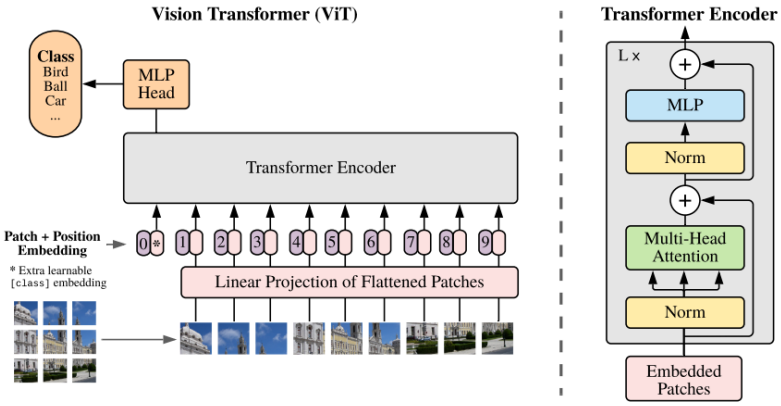

Figure 6: Overview of the Vision Transformer (ViT) architecture from (Dosovitskiy et al., 2021). The input image is split into patches, each flattened and linearly projected, with a learnable [class] token prepended. These embeddings are passed through a transformer encoder composed of stacked attention and feed-forward blocks.

The Vision Transformer (ViT) (Dosovitskiy et al., 2021) adapts the transformer architecture to image data by dividing an image into non-overlapping patches (e.g., 16×16), flattening them, and projecting each to a learnable embedding. These embeddings form a sequence processed by self-attention layers. Positional embeddings are added to retain spatial information.

This architecture was further adapted for generative modeling in DiT (Diffusion Transformer) (Peebles and Xie, 2023), which applies transformer blocks to patchified image representations in diffusion models. Key adaptations include:

- Patch-based tokenization of the input image.
- Time-step embeddings injected into the transformer.
- Optional conditioning via cross-attention for class or resolution control.

DiT has been shown to outperform convolutional backbones in certain diffusion settings (for example in the recent paper from Du et al. (2025)) and supports flexible conditioning mechanisms.

In our experiments, we used a DiT-like architecture for patchified wavelet generation, inspired by the K-Flow framework. Unlike K-Flow, which feeds all wavelet components (including noisy regions that we are not filling at the moment), we investigate selective token feeding to focus learning on the relevant resolution level.

2.6 Evaluation Metrics

Evaluating generative models that do not involve direct reconstruction—such as diffusion models or flow-based methods—requires distribution-level metrics. A commonly used metric in this context is the Fréchet Inception Distance (FID), which measures the similarity between the distribution of generated images and that of real images based on features extracted from a pretrained Inception network.

Given two sets of image features with empirical mean and covariance (μ_r, Σ_r) for the real data and (μ_g, Σ_g) for the generated samples, the FID is defined as:

$$\mathrm{FID}(r, g) = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}\right). \tag{21}$$

FID captures both the mean shift and the variance mismatch between the two distributions. Lower FID values indicate higher similarity. This metric is favored in generative modeling due to its correlation with human perception of image quality, even though it has known limitations:

- It relies on the Inception network, which may not align perfectly with all datasets or domains.
- It does not evaluate diversity or structural consistency at a fine level.
- It assumes Gaussian distributions in feature space, which may not strictly hold.

FID is known to be sensitive to subtle changes in image statistics, even when the perceptual quality remains high, as observed in studies on image compression and preprocessing of data sets (Parmar et al., 2022). Small, imperceptible changes, such as minor differences in texture, frequency content, or preprocessing artifacts, can significantly impact FID, even when images appear nearly identical to human observers. This is because FID measures distributional differences in the feature space rather than direct perceptual similarity. In particular, high-frequency details play a crucial role in its evaluation, meaning that images with slight deviations in fine textures or structural details can be penalized disproportionately. As a result, FID may not always accurately reflect perceptual quality, making it important to complement it with alternative metrics, such as perceptual similarity scores or human evaluations.

As also observed in our own experiments later in this report, increasing the number of function evaluations (NFE) typically reduces FID, suggesting better statistical alignment with the real data. However, a lower FID does not always reflect better perceptual quality. Excessive NFE can lead to artifact sharpening and other issues. This highlights the need to

complement FID with other evaluation techniques, such as perceptual similarity metrics or human assessment, especially when high-resolution details are involved.

In this report, we primarily rely on FID to quantify sample quality, due to its widespread adoption and ease of comparison across models. However, given its sensitivity and limitations—as discussed above—we also include qualitative visual assessments to better capture perceptual and structural fidelity. In particular, we highlight cases where lower FID does not always correspond to visually better results, especially when analyzing the impact of sampling parameters or high-frequency details.

While other alternative metrics offer valuable insights, this report primarily relies on the Fréchet Inception Distance (FID) for two main reasons. First, some metrics like LPIPS are most effective in reconstruction or image-to-image tasks where a specific ground-truth image exists for each generated output, a condition that does not apply to our unconditional generation setting. Second, and more importantly, FID remains the de facto standard for benchmarking in the vast majority of the generative modeling literature we compare against. To ensure a fair and direct comparison with the state-of-the-art results presented in our summary tables, we have adopted FID as our primary metric, as it is the most consistent measure available across prior work.

2.7 Comparative Summary

To better situate our approach within the broader landscape of generative modeling, we present a comparative summary of recent methods based on their reported FID scores and number of function evaluations (NFE). The values listed in the following tables are extracted directly from the literature and cover a variety of model types, including diffusion models, GANs, score-based methods, and Flow Matching variants. These benchmarks serve as a reference point for evaluating the trade-offs between sample quality and inference efficiency.

The following tables summarize performance on standard benchmarks. These include CIFAR-10, a dataset of 32x32 color images in 10 classes; CelebA, a large-scale dataset of celebrity faces; and CelebA-HQ, a high-quality version of CelebA. These datasets will be described in more detail in future sections.

Model	**FID**	**NFE**	**Type**	**Reference**
DDPM	7.48	274	Diffusion	Ho et al. (2020)
Score Matching	19.94	242	Score-Based	Vincent (2011)
ScoreFlow	20.78	428	Score-Based	Song et al. (2021a)
Flow Matching (FM)	8.06	183	Flow Matching	Lipman et al. (2023)
OT-Flow Matching	6.35	142	Flow Matching	Lipman et al. (2023)
EDM	1.97	35–1000+	Diffusion	Karras et al. (2022)
PD-EDM	9.12	1	Diffusion	Salimans and Ho (2022)
CD	5.83	2	Diffusion	Song et al. (2023)
StyleGAN-XL	∼2.0	—	GAN	Sauer et al. (2022)

Table 1: FID and number of function evaluations (NFE) for recent generative models On CIFAR-10. One-step and distilled models aim to minimize NFE while maintaining competitive FID.

Model	FID	NFE	Model Type	Reference
DDPM-IP @1000 steps	1.31	1000	Diffusion	Ning et al. (2023)
DDPM-IP @80 steps	2.67	80	Diffusion	Ning et al. (2023)
PDM-DDPM++	1.77	50	Diffusion	Wang et al. (2023)
Soft Diffusion (VE SDE + Blur)	1.85	∼280	Diffusion	Daras et al. (2022)
MMD-PMish-NAS	1.92	—	GAN	Reddy Pulakurthi et al. (2024)
DDPM++	2.90	1000	Diffusion	Kim et al. (2022)
Contrastive-DDPM	5.25	1000	Diffusion	Aneja et al. (2021)

Table 2: FID and sampling steps (NFE) for various generative models on CelebA (64×64).

Model	FID	NFE	Type	Reference
K-Flow (Wavelet-DiT L/2)	4.99	—	K-Flow Matching (DiT)	Du et al. (2025)
K-Flow (Fourier-DiT L/2)	5.11	—	K-Flow Matching (DiT)	Du et al. (2025)
K-Flow (PCA-DiT L/2)	5.19	—	K-Flow Matching (DiT)	Du et al. (2025)
ADM	5.82	85	Flow Matching (U-Net)	Dao et al. (2023)
FM	7.34	128	Flow Matching (U-Net)	Lipman et al. (2023)
LDM	5.11	50	Diffusion	Rombach et al. (2022)
LSGM	7.22	23	Score-Based	Vahdat et al. (2021)
WaveDiff	5.94	2	Diffusion (Wavelet)	Phung et al. (2023)
DDGAN	7.64	2	Hybrid (Diffusion + GAN)	Xiao et al. (2022)
Score SDE	7.23	4000	Score-Based Diffusion	Song et al. (2021b)
Consistency Flow Matching	36.4	6	Flow Matching	Yang et al. (2024)
FM with Learned Interpolants	28.6	6	Flow Matching	Shankar and Geffner (2025)

Table 3: FID and sampling steps (NFE) for generative models on CelebA-HQ (256×256).

2.7.1 Improving Fidelity vs. Reducing Inference Cost

A major axis of progress in generative modeling has focused on improving image fidelity, typically measured through the Fréchet Inception Distance (FID). High-fidelity models such as **EDM** (Karras et al., 2022), **StyleGAN-XL** (Sauer et al., 2022), and **DDPM-IP** (Ning et al., 2023) have set new baselines by refining noise schedules, introducing improved loss functions, and leveraging large-scale architectures. For instance, EDM explores architectural and training design spaces of diffusion models, while DDPM-IP proposes input perturbation strategies to reduce overfitting and improve generation sharpness. These methods often retain high numbers of denoising steps (e.g., 1000) but achieve top-tier realism, as reflected in FID scores below 2 on CelebA-HQ. In parallel, **score-based models** such as **Score SDE** (Song et al., 2021b) and **LSGM** (Vahdat et al., 2021) have integrated VAE-like latent structures or continuous-time sampling for better convergence in complex domains.

A complementary research direction has aimed at reducing the number of function evaluations (NFE) during inference, thus improving sampling speed. Methods like **Progressive Distillation (PD-EDM)** (Salimans and Ho, 2022), **Consistency Models (CD)** (Song et al., 2023), and **TRACT** (Berthelot et al., 2023) achieve impressive sampling efficiency by training models to perform denoising in one or two large steps. These approaches typically distill the behavior of slower teacher models into more aggressive student networks. Other innovations, such as **GET** (Geng et al., 2023) and **WaveDiff** (Phung et al., 2023), rely on architectural changes (e.g., equilibrium solvers, wavelet priors) to retain quality under minimal NFE. In this context, **Flow Matching** (Lipman et al., 2023) and its variants such as **OT-FM** provide a principled and efficient alternative to diffusion models, offering deterministic sampling in few steps, while maintaining strong FID performance. The emerging

class of **K-Flow** models (Du et al., 2025) further builds on this by leveraging scale-wise generation in wavelet, fourier or PCA domains.

While recent work such as Consistency Flow Matching (Yang et al., 2024) and Flow Matching with Learned Interpolants (Shankar and Geffner, 2025) explore novel training formulations, their current performance on CelebA-HQ remains below state-of-the-art, suggesting further refinement is needed. In contrast, the K-Flow approach from Du et al. (2025) reports surprisingly strong FID scores (below 5) using scale-wise tokenization strategies such as wavelet or PCA decomposition. However, as no code or models have been released at the time of writing, these results remain difficult to independently verify.

These two trends—fidelity-focused improvements and sampling-time reductions—are often at odds, with methods trading off one for the other. Recent works attempt to reconcile both, motivating hybrid or multi-scale generative architectures such as ours.

Summary

In summary, this review of the state-of-the-art has positioned Flow Matching as a powerful and efficient framework for generative modeling, offering a compelling alternative to slower diffusion-based approaches. To build a practical understanding of its behavior before designing our own architecture, the following section will provide a series of empirical experiments, visualizing the learned flows and pixel space structure of a baseline Flow Matching model.

3 Empirical Insights into Flow Matching

Before introducing our wavelet-based generative architecture, we present a series of experiments aimed at illustrating key behaviors of the Flow Matching (FM) framework. These experiments, conducted on MNIST and CIFAR-10, help build intuition about the learned transformations, latent space structure, and the effect of conditioning. Through visualization and qualitative analysis, we aim to clarify how FM differs from traditional diffusion models in practice, and why it serves as a strong foundation for hierarchical image generation.

3.1 Datasets and Experimental Setup

We conduct experiments using two datasets of increasing complexity to analyze the behavior of Flow Matching. The first is MNIST, a benchmark dataset of handwritten digits composed of 60,000 training and 10,000 test grayscale images, each of size 28×28. Due to its low dimensionality and visual simplicity, MNIST serves as an ideal environment for building intuition and visually interpreting the effect of learned probability flows. All images are normalized and resized as needed for consistency during training.

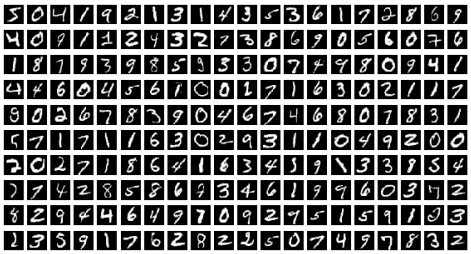

Figure 7: Sample images from the MNIST dataset. Each image is 28×28 and grayscale, representing a handwritten digit from 0 to 9.

The second dataset is CIFAR-10, a standard benchmark consisting of 60,000 32×32 RGB images spanning 10 object categories such as airplanes, cats, and trucks. In contrast to MNIST, CIFAR-10 presents higher complexity due to its color channels, background variability, and increased intra-class diversity. This allows us to assess how Flow Matching scales to more realistic, fine-grained data distributions. The same architectural backbone (a modified U-Net) is adapted to accommodate the multi-channel input and higher visual entropy of this dataset.

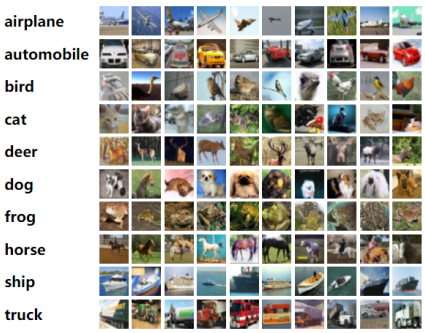

Figure 8: Sample images from the CIFAR-10 dataset. Each image is 32×32 RGB and belongs to one of 10 object classes.

3.2 Trajectory Visualization and Interpolation

Flow Matching defines a smooth probability flow from a simple prior distribution to a complex data distribution. To illustrate this behavior, we visualize the evolution of generated samples over time, starting from pure Gaussian noise.

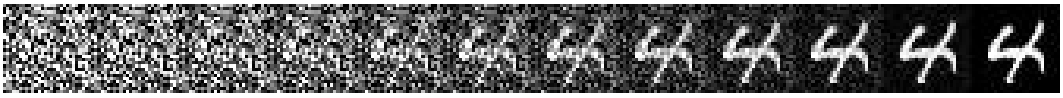

Figure 9: Trajectory of a sample transforming from Gaussian noise to a digit (MNIST). Each frame represents a later stage in the flow integration.

Figure 9 illustrates how a sample evolves from an initial noise vector toward a structured digit. Compared to stochastic denoising processes, Flow Matching yields a deterministic path that increasingly concentrates the sample toward the target data distribution over time.

To further demonstrate the continuity of the learned flow, we interpolate between two points in the initial latent space and observe how the generated images change.

For example, Figure 10 presents an interpolation sequence transitioning from a 4 to a 9, demonstrating the continuity of the learned distribution.

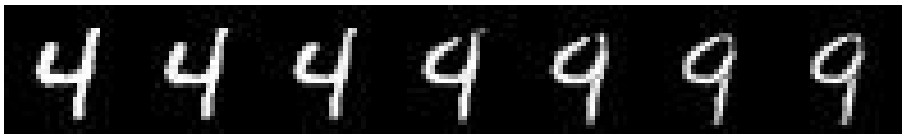

Figure 10: Latent interpolation between two noise vectors on MNIST. The generated digits evolve smoothly between two classes, with gradual shape and stroke changes.

The idea is that if X_0^4 and X_0^9 are two points in the initial gaussian distribution that respectively generate a 4 and a 9, then by applying the generation process to intermediate points obtained through **linear interpolation**:

$$X_0^\lambda = \lambda X_0^9 + (1-\lambda)X_0^4, \quad \lambda \in [0,1]$$

we can smoothly transition from a 4 to a 9 in a continuous manner. As λ varies from 0 to 1, the generated digit gradually rotates, stretches, and deforms, preserving a natural transformation between the two numbers.

It is important to be precise about the term 'latent interpolation.' The operation we perform is a simple linear interpolation between two vectors in the initial noise (latent) space. However, the resulting visual transition is a much more sophisticated phenomenon. As each point along this simple latent path is passed through the highly non-linear generative flow, it induces a corresponding non-linear path in the final pixel space. If the model has learned the data distribution well, this path travels smoothly along the learned data manifold. This is what creates the seamless and realistic transformation from one image to another, as opposed to a naive linear cross-fade which would produce blurry, unrealistic artifacts.

Figure 11: Latent interpolation between two noise vectors on CIFAR-10. The transformation progresses from a bird to a horse, with continuous changes in structure and texture.

Figure 11 shows the same process applied to CIFAR-10, interpolating between a bird and a horse. The transition is not only visually smooth, but semantically meaningful—colors,

shapes, and textures change continuously, confirming that the model has learned a well-structured latent space.

3.3 Latent Space Structure and PCA Analysis

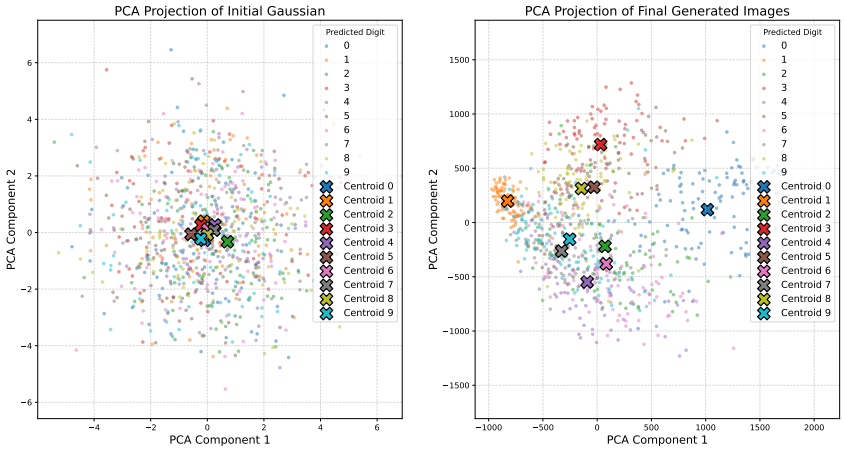

Figure 12: PCA visualization of the pixel space: (Left) The initial Gaussian distribution. (Right) The final structured distribution where digits are well-organized.

To analyze the structure of the learned distribution, we apply **Principal Component Analysis (PCA)** to the generated samples.

Figure 12 displays the PCA projection of a set of generated samples after Flow Matching. We can observe semantically coherent clusters, with visibly distinct centroids. Some digits remain closer (e.g., 5 and 8), while others are well separated (e.g., 0, 1, and 3), reflecting the structural relationships learned during training.

At first glance, the initial distribution projected in Figure 12 appears as a single, disorganized cluster of points where different classes are completely mixed. This is not a limitation of the model, but rather the expected and correct visualization of the starting noise distribution.

It is crucial to remember that we begin with samples from a standard isotropic Gaussian distribution in a high-dimensional pixel space (e.g., 784 dimensions for MNIST). 'Isotropic' implies that the distribution is perfectly symmetrical, with equal variance in all directions—like a high-dimensional sphere. It contains no inherent structure or preferred orientation that would separate points by a future class label.

This is where the role of PCA as an analysis tool becomes critical. PCA is a linear dimensionality reduction method that seeks to find the axes of maximum variance in the data. When applied to an isotropic Gaussian, where variance is the same in every direction, PCA cannot find any meaningful low-dimensional structure to represent. The resulting 2D projection is therefore a formless blob, which accurately reflects the unstructured nature of the initial noise. Points that happen to appear close in this 2D projection may, in fact, be very far apart in the original high-dimensional space.

Overall, the final distribution of the generated samples is more structured and organized than the initial Gaussian prior, confirming that Flow Matching effectively reshapes the initial distribution. The learned transformation preserves meaningful class-level relationships.

This structured geometry also explains why interpolations between noise vectors yield coherent image transitions, as intermediate points remain close to the data manifold throughout the flow, as seen in the previous section.

3.4 Conditioned Generation and Style Preservation

So far, we've seen how Flow Matching (FM) allows for the generation of samples. However, in generative models, we often seek **control over the output**, such as generating a sample from a specific class in MNIST or guiding text-to-image generation.

This can be achieved through **conditioning**, where additional information is provided to influence the generation process. Instead of modeling an unconditional distribution $p(X)$, we introduce a conditioning variable y (e.g., a class label or a text prompt) and model the **conditional distribution**:

$$p(X|y)$$

where y provides guidance to the model. In the context of Flow Matching, this is done by modifying the vector field estimation to incorporate y typically through label embedding concatenation or cross-attention, depending on the model architecture, effectively steering the learned trajectories toward the desired output. This allows the models to focus on the other attributes of the data.

3.4.1 Class-Conditional Flow Matching

In the case of **digit generation** with MNIST, we can condition the model on a class label $y \in \{0, 1, \ldots, 9\}$, ensuring that the generated sample belongs to the specified digit category.

In this section, we present the results of applying Flow Matching to MNIST, highlighting the evolution of digit generation, variations along PCA directions, and conditional denoising.

3.4.2 Digit Variation Along PCA Axes

To analyze how digits change along meaningful directions in the latent space, we visualize the variation of digits 4, 5, and 9 along their **PCA principal components**. Each row in Figure 13 represents samples generated by shifting a latent vector along a major PCA direction.

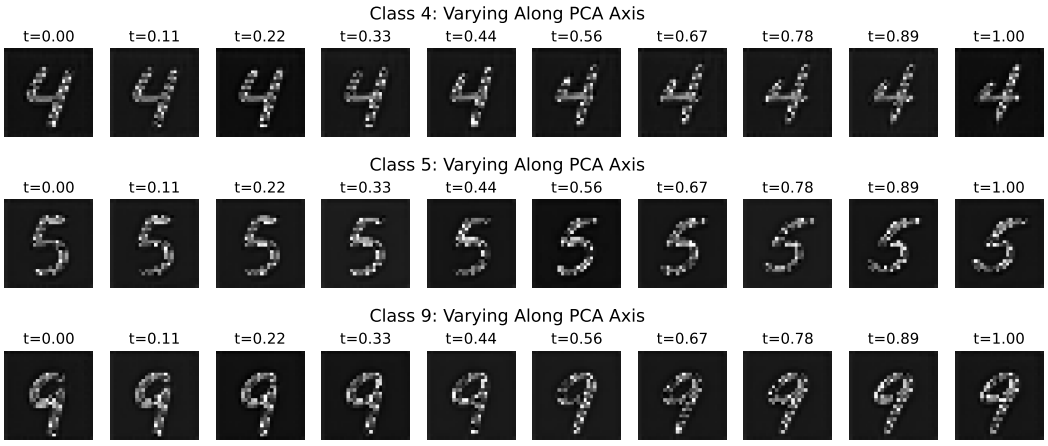

Figure 13: Variations of digits along their principal PCA directions. Each row shows the effect of moving in the pixel space along a principal component vector. **Note: Here, t represents the interpolation coefficient along the PCA axis and is not related to the time variable t of the flow matching process.**

3.4.3 Conditional Denoising and Style Preservation

Flow Matching allows **conditioning** during generation, meaning we can guide the model to produce different digits from the **same initial noise sample**, while maintaining a **con-**

sistent handwriting style. This is demonstrated in denoising images (Figures 14.

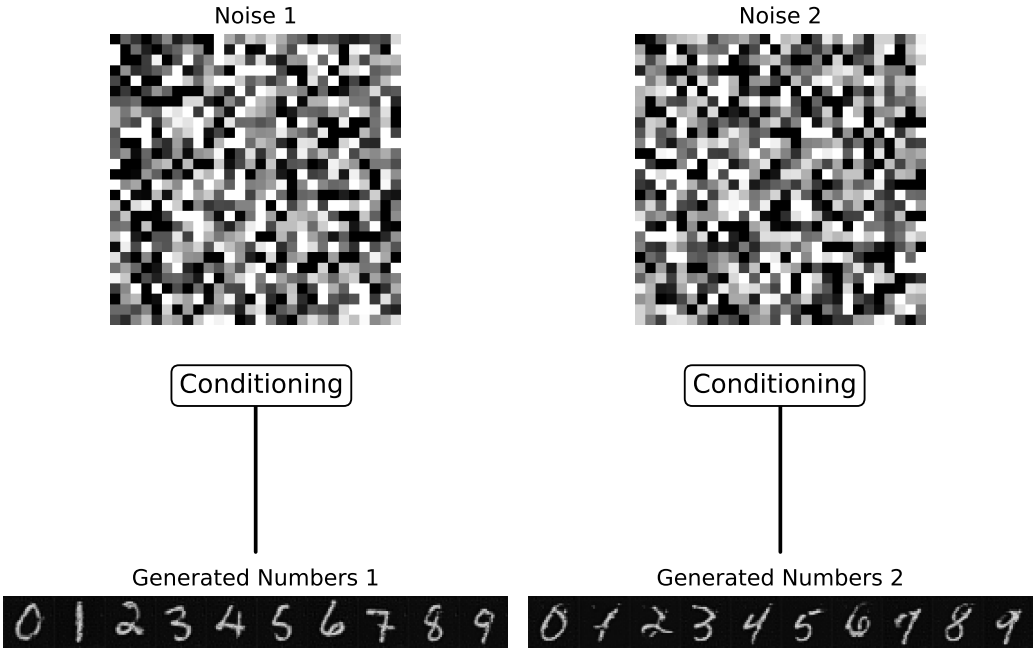

Figure 14: Conditional denoising process. The model generates different digits from the same noise, maintaining consistent writing style.

The first set has slightly more **rounded** numbers—particularly visible in digits like **5, 6, and 9**. In contrast, the second set appears more **upright**, with straighter and sharper digit formations, which also appear **tilted to the right**. Additionally, numbers like **4 and 7** exhibit distinct calligraphic styles between the two sets.

Without conditioning, the model must learn both **class structure** and **handwriting style**. By conditioning on the class, the model is explicitly guided toward generating a specific digit, allowing it to focus purely on stylistic consistency. This enables the model to preserve key handwriting attributes across different digits, such as **Stroke thickness**, **Slant**, **Curvature** or **Writing consistency**.

3.4.4 Generalizing to Other Modalities

Beyond image generation guided to a given class, conditioning extends to various domains:

- **Text-to-image generation**: where y is a text prompt (e.g., Stable Diffusion).
- **Style-conditioned music generation**: where y represents a genre or mood.
- **Image denoising/deblurring**: where y is a noisy image that we want to denoising/deblurring.

By incorporating conditioning, Flow Matching can be extended to **controlled generative tasks**, making it more versatile in practical applications.

3.5 Performance Metrics and Qualitative Samples

To evaluate the quality of generated images, we compute the Frechet Inception Distance (FID) for different models and steps using torchmetrics.

Table 4: FID scores for Flow Matching (FM) and Optimal Transport Flow Matching (OT-FM) on CIFAR-10 at different Numbers of Function Evaluations (NFE). Evaluated using torchmetrics.

NFE	OT-FM, 200k steps	OT-FM, 400k steps	FM, 200k steps
1	240.47	231.03	355.7
2	89.86	93.93	160.03
5	23.24	22.81	34.96
10	13.18	12.35	14.41
50	6.45	5.73	6.80

The table 4 presents FID scores for different Flow Matching (FM) and Optimal Transport Flow Matching (OT-FM) models at varying Numbers of NFE. We observe that increasing NFE leads to significantly lower FID scores, indicating improved image quality. Furthermore, OT-FM consistently outperforms standard FM, particularly at lower NFE, which is consistent with the results observed in the 2D distribution experiments.

3.6 Qualitative Results on CIFAR-10

To complement the quantitative metrics, we also showcase a few generated samples. Figure 15 displays the images synthesized by the trained OT-FM model at 400k steps. We observe that:

- **Visual Fidelity:** The generated images exhibit improved sharpness and diversity as training progresses.
- **Variety of Classes:** Different classes (e.g., airplanes, dogs, cars) are clearly recognizable, indicating that the model captures key class-specific features.
- **Impact of NFE:** As shown in Figure 16, increasing the number of function evaluations (NFE) makes the generated samples sharper and more detailed.

Figure 15: CIFAR-10 samples (400k steps) with a NFE of 50.

(a) NFE = 2

(b) NFE = 5

(c) NFE = 10

(d) NFE = 50

Figure 16: Generated images at step 400k for different NFE values for the same initial noises.

Summary

Through these empirical investigations, we have demonstrated that Flow Matching effectively learns a continuous and structured mapping from noise to data on benchmark datasets. The latent space visualizations and conditioning experiments confirm its capabilities, providing a solid foundation for our work. However, these standard implementations also highlight the inherent challenge of scaling directly to high resolutions. Therefore, building on these insights, the next section introduces our core contribution: a novel multi-scale architecture that leverages wavelet decomposition to address this challenge.

4 Multi-Scale Image Generation

Building on the insights gained from Flow Matching on MNIST and CIFAR-10, we now explore how to extend this approach to higher-resolution image generation using a multi-scale framework. Instead of generating full-resolution images in one pass, we decompose the process into low-frequency and high-frequency components using wavelet transforms. This separation allows us to design more efficient and modular generative pipelines, capable of scaling to higher resolutions while maintaining coherence and sample quality.

4.1 Wavelets for Generative Modeling

Wavelets are mathematical functions that decompose signals into different frequency components, with each component studied at a resolution matched to its scale (Mallat, 2008). Unlike Fourier transforms which use sine and cosine functions that extend infinitely, wavelets are localized in both time and frequency domains, making them particularly useful for analyzing signals with discontinuities or sharp transitions.

Formally, a wavelet is a function $\psi(t) \in L^2(\mathbb{R})$ that satisfies the admissibility condition:

$$C_\psi = \int_{\mathbb{R}} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty \tag{22}$$

where $\hat{\psi}(\omega)$ is the Fourier transform of $\psi(t)$. This condition implies that $\int \psi(t)dt = 0$, meaning the wavelet must oscillate.

From a mother wavelet $\psi(t)$, a family of wavelets is generated through scaling and translation:

$$\psi_{a,b}(t) = |a|^{-1/2} \psi\left(\frac{t-b}{a}\right) \tag{23}$$

where a is the scaling parameter and b is the translation parameter.

Wavelets in Image Processing

For image processing, wavelets provide a powerful framework for multi-resolution analysis. A 2D wavelet transform decomposes an image into a set of frequency bands, capturing horizontal, vertical, and diagonal details at different scales.

When applied to an image, the discrete wavelet transform (DWT) produces four subbands:

- LL (Low-Low): Approximation coefficients (an average filter or downsampled version of the image).
- LH (Low-High): Horizontal detail coefficients (a vertical edge detector).
- HL (High-Low): Vertical detail coefficients (a horizontal edge detector).
- HH (High-High): Diagonal detail coefficients (a diagonal edge detector).

The 2D DWT is typically implemented using separable filtering: first applying 1D wavelet transforms along rows, then along columns. For an image I, the process can be expressed as:

$$\text{DWT}(I) = \{LL, LH, HL, HH\} \tag{24}$$

The LL subband is essentially a downsampled, coarser version of the original image, while the other subbands capture different directional details. This decomposition can be applied recursively to the LL subband, creating a multi-level representation.

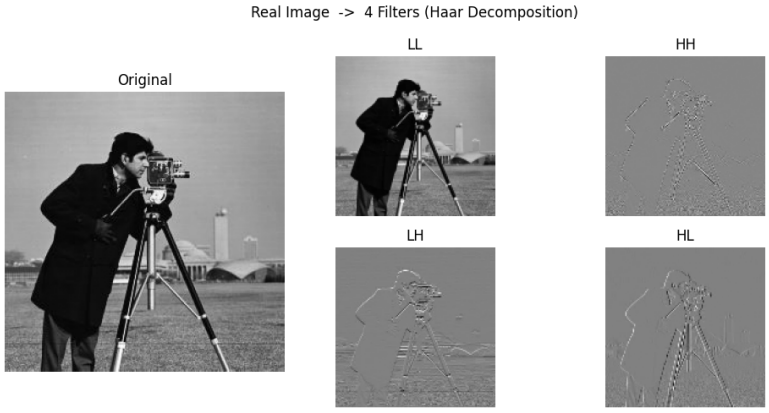

Figure 17: Wavelet decomposition of an image showing the LL (approximation), LH (horizontal detail), HL (vertical detail), and HH (diagonal detail) subbands.

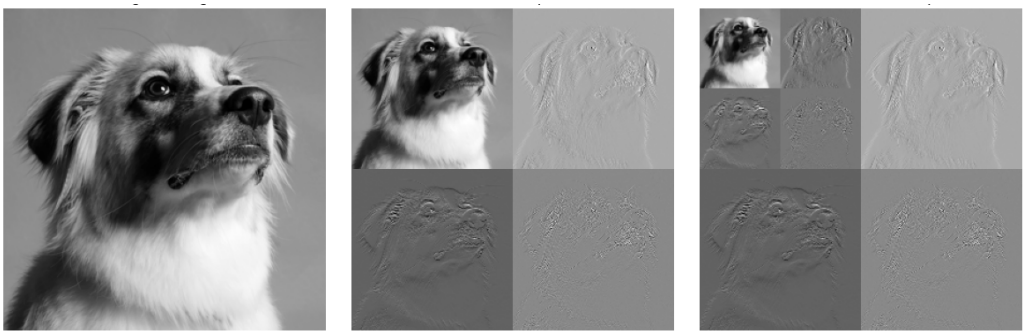

Figure 18: Two-level Haar wavelet decomposition of an image. Left: original grayscale image. Center: first-level decomposition into low-frequency (LL_1) and high-frequency (LH_1, HL_1, HH_1) components. Right: second-level decomposition, where LL_1 is itself decomposed into LL_2 and high-frequency bands (LH_2, HL_2, HH_2).

While the decomposition structure into LL, LH, HL, and HH subbands is shared across many wavelet families, the exact appearance and smoothness of the subbands depend on the choice of wavelet basis. In our case, we use the Haar wavelet due to its simplicity and efficiency.

The Haar Wavelet

The Haar wavelet is the simplest wavelet, defined as:

$$\psi(t) = \begin{cases} 1 & \text{for } 0 \leq t < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

Its scaling function is:

$$\phi(t) = \begin{cases} 1 & \text{for } 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

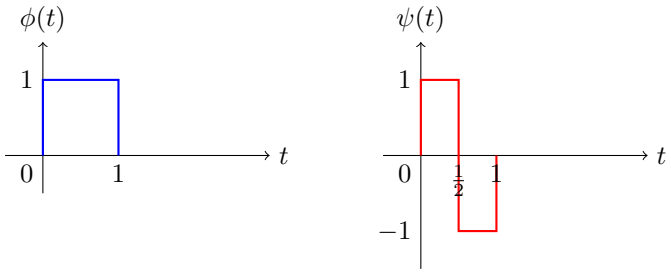

Figure 19: The Haar scaling function $\phi(t)$ and wavelet function $\psi(t)$.

For image processing, the Haar transform applies simple operations: averaging and differencing. For a 1D signal $[a, b]$, the transform produces:

$$\text{Approximation} = \frac{a+b}{\sqrt{2}} \tag{27}$$

$$\text{Detail} = \frac{a-b}{\sqrt{2}} \tag{28}$$

When applied to images, the 2D Haar transform filters rows and columns with the following operations:

$$LL = (a+b+c+d)/2 \tag{29}$$

$$LH = (a-b+c-d)/2 \tag{30}$$

$$HL = (a+b-c-d)/2 \tag{31}$$

$$HH = (a-b-c+d)/2 \tag{32}$$

where a, b, c, d are the four pixels in a 2×2 block.

The Haar wavelet is particularly advantageous for your flow matching model due to:

- Simplicity: The transform is computationally efficient
- Perfect reconstruction: The original image can be exactly recovered
- Edge detection: It naturally captures horizontal, vertical, and diagonal edges
- Locality: Each coefficient relates to a specific spatial location

By generating the LL filter first and using it as conditioning, your approach leverages the hierarchical nature of the wavelet transform, allowing the generative model to focus on the overall structure (captured in LL) before adding finer details (in LH, HL, and HH).

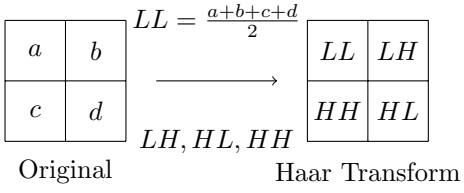

Figure 20: Illustration of the Haar wavelet decomposition.

4.2 Method Overview

Our method is inspired by the hierarchical generation paradigm explored in Guth et al. (2022), where **wavelets** were combined with **Score-Based Generative Models (SGMs)** to improve efficiency in diffusion models. In contrast, we explore this idea in the context of **Flow Matching**, which provides an alternative continuous-time generative modeling framework.

Our approach consists of the following steps:

- We first decompose an image into four wavelet bands using **Discrete Wavelet Transform (DWT)**: the **low-frequency (LL)** component and three **high-frequency (HF)** components (**HL**, **LH**, **HH**).
- We train a first **Flow Matching** model to generate the **LL band** from Gaussian noise.
- A second **Flow Matching** model is trained independently to generate the **HF bands** (**HL, LH, HH**), conditioned on the generated LL.
- Finally, we apply the **Inverse Wavelet Transform (IWT)** to reconstruct the full-resolution image from the generated LL and HF components.

Unlike traditional diffusion or flow models that generate full-resolution images in one step, our pipeline decouples structure and detail generation. This hierarchical approach ensures that the model first captures the coarse structure of the image before refining the details, leading to more stable and high-quality generation.

Figure 21 illustrates the proposed architecture. The **LL band** is generated first, followed by the **HF bands**, which are conditioned on the LL. The final image is reconstructed via IWT.

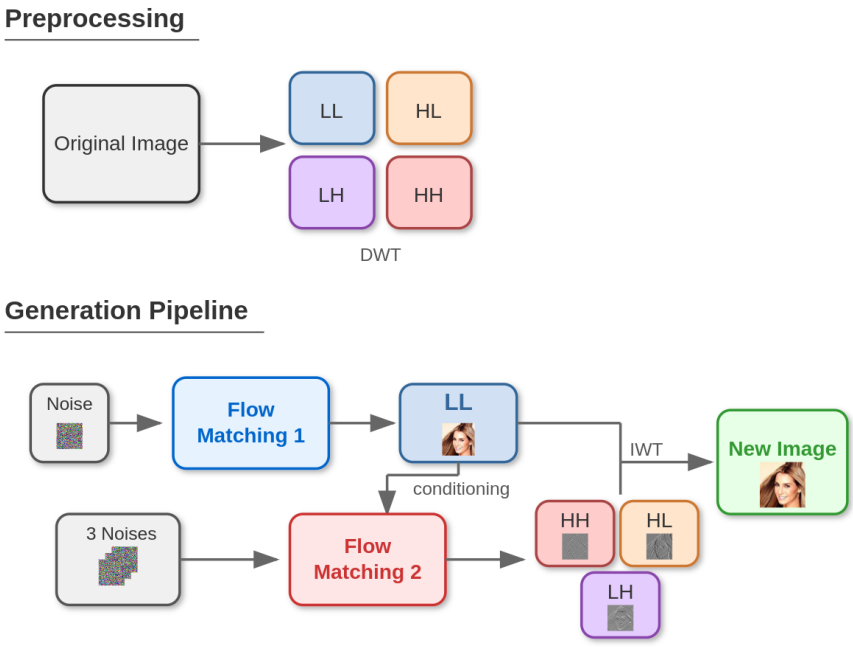

Figure 21: Overview of the proposed Wavelet-Based Flow Matching architecture. The model first generates the LL component from Gaussian noise, then the HF components conditioned on LL, and finally reconstructs the image via IWT.

4.3 Single-Scale Architecture

4.3.1 Results

To evaluate the effectiveness of our model, we compute the **Fréchet Inception Distance (FID)** for different configurations of LL and HF generation.

Table 5: FID scores for different LL and HF configurations. All images were generated using NFE = 20 for the first and second models

FID Index	Generation Method	LL	HF	FID Value
FID1	Full pipeline	Generated	Generated	27.85
FID2	LL-only model without HF	Generated	None	98.41
FID3	Real LL images without HF	Real	None	63.48
FID4	Real LL images with generated HF	Real	Generated	3.92

Table 6: FID score for LL-only generation comparison between Real LL and Generated LL at NFE = 20.

FID Index	Generation Method	FID Value
FID5	LL-only model	14.13

Table 7: FID scores and generation time per batch of size 32 for different NFE values.

NFE	FID Score	Time (sec/batch)
5	71.76	0.1213
10	45.55	0.1668
30	23.72	0.2462
50	18.36	0.3469
100	17.33	0.6234
200	15.34	1.2076
300	14.21	1.6016

Table 8: Model parameter count for LL and HF generators.

Model	Number of Parameters
Model 1 (LL generator)	150,242,307
Model 2 (HF generator)	74,072,076
Total	**224,314,383**

4.3.2 Analysis and Discussion

From the results, we observe the following key insights:

- **HF components significantly improve generation quality**:
  - **FID2 (98.41)** vs. **FID4 (3.92)** shows that a generated LL component alone is insufficient. Adding generated HF details drastically improves realism.

    - Even when using a real LL component, **FID3 (63.48)** is much higher than **FID4 (3.92)**, reinforcing that HF details are crucial.
- **Flow Matching 1 generates reasonable LL structures, but improvement is possible**:
    - **FID5 (14.13)** indicates that the LL-only model produces decent low-frequency components.
    - However, since **FID1 (27.85)** is worse than **FID4 (3.92)**, we infer that the quality of the **LL components is a bottleneck in the overall pipeline.**
- **The final pipeline outperforms naive upscaling of real images**:
    - **FID1 (27.85)** is much better than **FID3 (63.48)**, confirming that our full approach (LL + HF generation) significantly outperforms naive IWT-based upscaling without HF.
    - However, further improvements in Flow Matching 1 (LL generation) could enhance the final image quality.

While these results highlight clear trends in architectural effectiveness, it is also important to reflect on the limitations of the FID metric itself.

FID is known to be sensitive to subtle changes in image statistics, even when the perceptual quality remains high, as observed in studies on image compression and preprocessing of data sets Parmar et al. (2022). Small, imperceptible changes, such as minor differences in texture, frequency content, or preprocessing artifacts, can significantly impact FID, even when images appear nearly identical to human observers. This is because FID measures distributional differences in the feature space of a pretrained model, rather than direct perceptual similarity. In particular, high-frequency details play a crucial role in its evaluation, meaning that images with slight deviations in fine textures or structural details can be penalized disproportionately. As a result, FID may not always accurately reflect perceptual quality, making it important to complement it with alternative metrics, such as perceptual similarity scores or human evaluations.

As shown in Table 7, increasing the number of function evaluations (NFE) consistently reduces the FID score, suggesting a better alignment with the data distribution. However, a lower FID does not always translate to better perceptual quality. In practice, a high NFE can lead the model to over-optimize, sharpening imperfections or artifacts in the generated images, highlighting the limitations of FID as a sole metric for perceptual assessment.

4.3.3 Qualitative Samples

While FID provides a quantitative measure of similarity between distributions, it does not always reflect perceptual quality—especially in cases where minor high-frequency differences can disproportionately affect the score. To address this, we present qualitative samples that illustrate the visual effects of our wavelet-based architecture. These examples help assess perceptual realism, structural coherence, and the impact of high-frequency reconstruction beyond what numerical metrics alone can capture.

Figures 22 and 23 illustrate the impact of high-frequency (HF) reconstruction on image quality by comparing different reconstruction methods.

Figure 22 presents fully generated images, highlighting the difference between IWT upscaling without HF (left) and LL with generated HF (right). The left column, which lacks HF components, appears significantly blurry, with a noticeable loss of fine details such as facial textures and hair strands. The right column, where HF has been generated, exhibits sharper features and improved perceptual quality, demonstrating that

HF details are essential for realism in fully generated images, as previously observed during the analysis of the FID values.

Figure 22: Comparison of generated LL images with IWT without the HF (left) vs. Fully generated images using LL and HF (right).

Figure 23, on the other hand, presents a comparison using real images, focusing on the effect of the IWT with HF components.

- The **top row** consists of real unaltered images.
- The **middle row** contains LL components of real images, upscaled using IWT without the HF components, resulting in a loss of fine details and producing softened and lower quality versions of the originals.
- The **bottom row** reconstructs the real images by combining the LL component with the generated HF. Interestingly, the generated HF appears to sharpen and refine the real images, in some cases making them look even clearer than the originals by removing noise and enhancing fine structures.

This interesting phenomenon suggests that the HF generator doesn't simply learn to add random texture; it learns a powerful prior for natural high-frequency details. When conditioned on a real LL image, it effectively acts as a blind "super-resolution" or "refinement" filter, suppressing noise from the original image and synthesizing a canonical, clean texture. This suggests a potential application for the HF model as a standalone image enhancement tool.

Figure 24 illustrates smooth transitions between two latent noise vectors, showcasing continuous changes in multiple facial attributes, such as gender, skin tone, facial expression, and head orientation.

These results highlight the model's ability to interpolate smoothly while maintaining perceptual realism.

Figure 23: Comparison of reconstruction methods on images from the CelebA dataset. Top row: Real, unaltered images. Middle row: Real LL images reconstructed via IWT without HF components, showing blurriness. Bottom row: Real LL images combined with generated HF components, showing enhanced sharpness and detail..

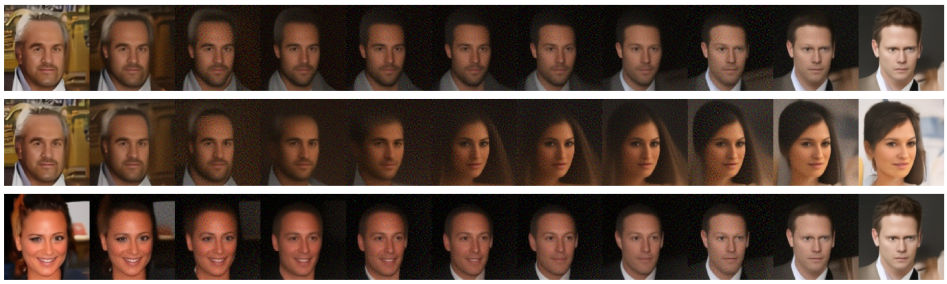

Figure 24: Pixel space interpolations on the data manifold. Each row shows a smooth transition between two images in the pixel space, demonstrating the model's ability to preserve structure and realism across facial attributes.

4.3.4 Benefits of the Wavelet-Based Architecture

- **Reduced Computational Cost**: By first generating a lower-resolution LL component before refining details with HF components, the model requires fewer computations compared to directly generating a high-resolution image. This significantly reduces both training and inference costs.
- **Faster Training & Fine-Tuning**: Since the LL and HF models can be trained separately, each network can be smaller and more efficient. This allows for faster convergence and makes fine-tuning more practical, as improvements can be made to the HF generator without retraining the entire pipeline.
- **Data Scaling**: The hierarchical nature of this method enables iterative application, where a small LL is generated first, followed by progressively finer HF components. This approach allows for multiple applications of the Inverse Wavelet Transform (IWT), making it well-suited for scalable image synthesis.
- **Modular and Flexible Design**: The separation of LL and HF components makes the architecture highly modular. Individual components can be improved or replaced independently, allowing for easier adaptation to different datasets, resolutions, or model refinements without retraining the entire system.

One practical strength of our architecture is that the U-Net-based HF generator can generalize beyond its training resolution. Although trained on 32×32 images, it can be applied to larger inputs at inference time, enabling recursive upsampling through repeated application of the inverse wavelet transform (IWT). Remarkably, despite being trained solely on face images (CelebA), the HF model also generalizes to other domains such as landscapes and animals, suggesting that it captures reusable high-frequency priors rather than overfitting to a specific class of images.

4.4 Multi-Scale Architecture

Building on these observations, we propose a scalable two-stage architecture for image synthesis. The process begins with a low-resolution LL generator trained on a specific dataset (e.g., CIFAR-10, CelebA), responsible for producing a coarse structural prior. A second HF generator is then trained separately across multiple resolutions, conditioned on the LL image and the target output scale.

This design enables recursive image generation: starting from a small LL image, high-frequency details are successively added at increasing resolutions using IWT, allowing the model to scale flexibly and efficiently to higher dimensions. Training the HF generator on a large, diverse dataset ensures domain generalization, making the overall pipeline both modular and resolution-agnostic. The next section describes the implementation and experimental results of this recursive framework.

To train the high-frequency (HF) generator, we selected the COCO 2017 dataset (Lin et al., 2015) due to its high visual diversity and broad coverage of natural image statistics. Unlike domain-specific datasets such as CelebA, COCO contains a wide range of objects, textures, and scenes—including animals, vehicles, indoor environments, and outdoor landscapes. This diversity helps the HF generator learn general high-frequency patterns that are not tied to a specific category, improving its ability to generalize across domains when conditioned on LL components from various sources.

4.4.1 Results

We evaluate the performance of our multi-scale pipeline across multiple axes to measure both reconstruction accuracy and generative quality at different stages and resolutions. Specifically, we report:

– **High-Frequency Reconstruction on COCO**: To assess the HF generator's ability to reconstruct natural details from real images, we compute the FID between original COCO images and reconstructed ones obtained by combining real LL components with generated HF components. This is done at resolutions 32×32, 64×64, 128×128, and 256×256. The model was trained on COCO 2017 due to its high variability and richness in content (e.g., people, animals, backgrounds), which encourages generalization across domains.

– **LL Generation on CIFAR-10 and CelebA-HQ**: We measure how well the LL generator captures global low-frequency structure by comparing generated 16×16 LL images to the true LL wavelet components extracted from CIFAR-10 and CelebA-HQ datasets. The small resolution ensures low computational cost, and retraining is fast (e.g., less than 2 hours on a single GPU).

– **Full Pipeline Results on CIFAR-10**: We evaluate the full generative pipeline by computing FID between original 32×32 CIFAR-10 images and reconstructions obtained via:

  * a generated LL component + HF generated by a model trained on **COCO** (cross-domain generalization),
  * a generated LL component + HF trained directly on **CIFAR-10** (domain-specific tuning).

– **Number of Parameters and Inference Cost**:

  * LL Generator: 120,062,128 parameters, NFE = 20
  * HF Generator: 80,072,076 parameters, NFE = $15 \times n_{\text{scales}}$ (e.g., 45 for 3 upsampling steps)

  The modular separation between LL and HF generators enables faster training and inference while allowing scalable resolution through recursive application of the IWT.

Table 9: FID for reconstruction on COCO using real LL and generated HF at different resolutions.

Resolution	**FID (Real LL + Generated HF)**
32×32	4.8310
64×64	3.8431
128×128	3.4922
256×256	3.2413

Table 10: FID and number of function evaluations (NFE) for different generation strategies on CIFAR-10. While our modular LL+HF pipeline yields slightly higher FID than standard FM and OT-FM, it achieves comparable visual quality with significantly fewer function evaluations. These baseline scores are reported at 50 NFE as in the original experiments. While a direct comparison at 35 NFE is not available, based on the trend observed in Table 4, their FID scores at 35 NFE would likely be higher than those reported here, reinforcing the efficiency of our proposed method.

Configuration	**FID**	**NFE**
Wavelet Transform Loss (32x32)	0.43	-
OT FM (direct 32×32 generation)	5.73	50
Standard FM (direct 32×32 generation)	6.80	50
LL only (generated LL vs. true LL)	4.95	20
LL + HF (HF trained on COCO) (32x32)	8.23	20 + 15 = 35
LL + HF (HF trained on CIFAR-10) (32x32)	7.95	20 + 15 = 35

Table 11: FID and number of function evaluations (NFE) for our multi-scale generation pipeline on CelebA-HQ at 256×256 resolution. The full pipeline generates HF components recursively across multiple scales.

Configuration	**FID**	**NFE**
LL only (generated LL vs. true LL at 16×16)	13.7	20
Full pipeline (LL+HF recursively applied up to 256×256 resolution)	35.2	20 + 15 × 4 = 80

4.4.2 Analysis and Discussion

We derive several insights from our experiments across different datasets and configurations:

- **LL Generation as a Bottleneck**: On both CIFAR-10 and CelebA-HQ, the FID obtained from LL-only generation is significantly higher than that of the full pipeline using real LL inputs. This highlights that the low-frequency component (LL) remains the main bottleneck of our architecture. Despite being efficient to train and generate, it limits the final quality.
- **Error Accumulation in Recursive HF Generation**: On CelebA-HQ, the recursive generation up to 256×256 results in a sharp increase in FID to 35.2. This quantitative degradation reflects the qualitative artifacts observed in Figure 26, where imperfections present in early stages become more pronounced at higher resolutions. This confirms that minor errors in the flow can compound at each upsampling step, highlighting a key challenge in recursive generative frameworks.
- **Efficiency vs. Performance Trade-off**: Compared to direct 32×32 generation with Flow Matching (FM) or Optimal Transport FM (OT-FM), our modular LL+HF pipeline achieves comparable FID scores with significantly lower inference cost (e.g., 35 NFE vs. 50). While the performance is slightly lower in terms of FID, the architectural modularity enables better scalability and faster inference.
- **HF Generator Generalization Across Domains**: The high-frequency generator trained on COCO achieves similar performance when used on CIFAR-10, showing a degree of domain generalization. However, slight improvements are obtained when training the HF generator directly on the target dataset, suggesting that domain-specific tuning still provides measurable gains.

– **Scale-dependent Learning of HF Components**: The FID scores on COCO reconstruction reveal that HF components at higher image resolutions (e.g., 128×128, 256×256) are easier to learn than those at lower resolutions (e.g., HF at 16×16, used to produce 32×32 images). This can be attributed to the fact that low-scale HF components still carry semantically significant information—such as edges, contours, and global texture—which requires more complex modeling. In contrast, HF components at higher scales predominantly contain fine-grained local textures, which are easier to synthesize.

4.4.3 Qualitative Samples

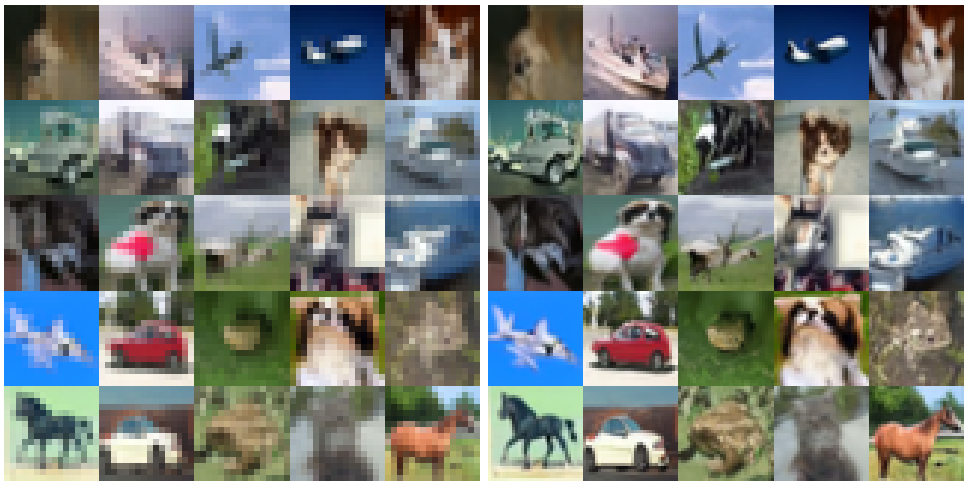

Figure 25: Comparison of generated CIFAR-10 images. **Left:** Low-frequency (LL) images at 16×16. **Right:** Final 32×32 images reconstructed using generated LL and high-frequency components (HF), combined via inverse wavelet transform (IWT).

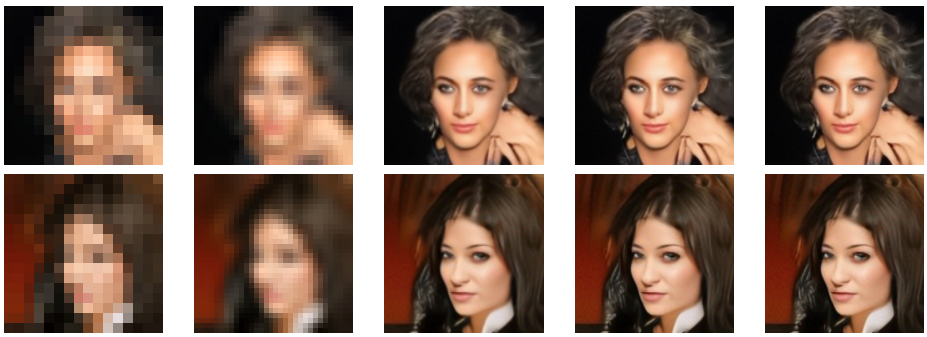

Figure 26: Progressive image reconstruction across resolutions for two subjects. Left to right: 16×16, 32×32, 128×128, 256×256, and 512×512.

Qualitative Evaluation. We provide visual examples to assess the reconstruction quality of our wavelet-based generation pipeline. Figure 25 compares generated CIFAR-10 samples using only the low-frequency (LL) component at 16×16 resolution (top row) with the final 32×32 images reconstructed after adding high-frequency (HF) components through the inverse wavelet transform (IWT) (bottom row). While the LL-only images retain basic semantic structure, they lack detail and appear blurry.

The addition of HF components significantly enhances texture, edges, and sharpness, producing coherent and realistic outputs.

Figure 26 shows progressive image reconstruction across resolutions for two CelebA-HQ samples. Each column corresponds to an increasing resolution level, from 16×16 up to 512×512. At low resolutions, the model captures identity and global facial structure; as resolution increases, fine-grained features such as eyes, hair, and skin texture become more detailed and realistic. While this demonstrates the model's ability to preserve semantic consistency and enhance perceptual quality at each recursive scale, it also reveals a limitation: imperfections or artifacts present in earlier stages may become more pronounced as resolution increases, sometimes resulting in exaggerated textures or unnatural features. These examples highlight both the strength and the sensitivity of our modular generation pipeline, confirming that decomposing the process into LL and HF stages offers scalability and control, but also calls for careful refinement at each level.

4.4.4 Benefits of the Multi-Scale Architecture

Compared to the single-scale approach, our multi-scale pipeline provides both practical and performance improvements:

- **Resolution-Conditioned HF Generator:** Unlike the previous model, the HF generator in this pipeline is explicitly trained with resolution as a conditioning signal. This enables it to adapt its outputs to different target resolutions, improving detail synthesis at each scale and allowing more consistent performance from 32×32 up to 256×256.
- **General-Purpose HF Generator:** Once trained on a large, diverse dataset (e.g., COCO), the HF model can be reused across domains and resolutions. This decoupling allows the LL generator to be retrained on small or domain-specific datasets (e.g., CIFAR-10, CelebA-HQ) while retaining high-quality detail synthesis from the shared HF model.
- **Modularity and Dataset Flexibility:** The architecture permits easy substitution of the LL generator. For instance, one can use a lightweight LL model for fast inference or a more expressive one for higher fidelity, without retraining the HF generator. This modularity streamlines adaptation to new tasks.
- **Anytime Inference:** Because the image is constructed recursively, inference can be interrupted at intermediate stages (e.g., 64×64), allowing dynamic tradeoffs between quality and speed.

While our pipeline achieves an FID score comparable to baseline methods, its key advantage lies in its modularity, which unlocks a higher degree of flexibility and control. This design allows for a targeted approach to improvement: one can focus on enhancing either the LL generator to fix the structural bottleneck or the HF generator to refine details, without retraining the entire system. Furthermore, this separation grants granular control over the generation process at inference time. A user can independently modulate the Number of Function Evaluations (NFE) for the coarse and detail stages, enabling a dynamic trade-off between speed and quality. Most importantly, it is this modular architecture that enables a truly flexible, multi-scale generation pipeline, where the HF generator can be recursively applied to progressively upscale an image to various target resolutions—a feat impractical for traditional models.

4.5 Theoretical Analysis of K-Flow Token Efficiency

Recent work on K-Flow (Du et al., 2025) introduced a powerful coarse-to-fine generation framework. A careful analysis of the official architecture, however, reveals a

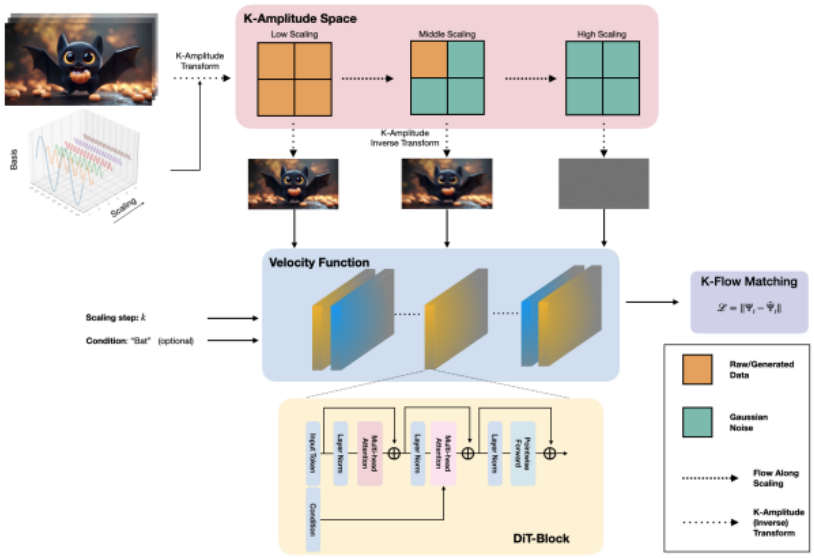

Figure 27: The K-Flow architecture from Du et al. (2025) . In this framework, noise is added in the frequency domain, but an inverse transform returns the data to the pixel space before the DiT backbone learns the velocity function.

key design choice: the main DiT (Diffusion Transformer) backbone operates in the pixel space. As illustrated by the authors' diagram in Figure 27 , the process involves applying a K-Amplitude transform (e.g., wavelets), adding noise at different scales in this frequency domain, and then performing an inverse transform to return to the pixel space. It is this resulting noisy image that is then patchified, tokenized, and fed to the DiT to learn the velocity field.

In our work, we investigate an alternative architectural design aimed at working directly in the wavelet domain. This approach avoids the need for a constant back-and-forth transformation to pixel space and may allow the model to learn the statistics of frequency components more directly.

At each level s, the input consists of a sequence of **patchified wavelet bands**, split into tokens and processed by a DiT (Diffusion Transformer) model. Each wavelet band (LL, LH, HL, HH) is decomposed into $k \times k$ spatial patches, resulting in k^2 tokens per band. At scale $s = 0$, only the LL band is present; for higher scales $s > 0$, the three HF bands (**LH**, **HL**, **HH**) are added, one triplet per level.

After applying the multi-level wavelet transform to the data, we patchify the wavelet coefficient tensors themselves. From here, we propose and analyze two distinct strategies for feeding these wavelet-space tokens to a DiT backbone:

To prepare the input for the DiT-based K-Flow model, we first apply the discrete wavelet transform (DWT) recursively up to a target level $s_{\max}$. This produces a hierarchy of wavelet bands: one LL (low-frequency) component at level 0, and three high-frequency bands (LH, HL, HH) at each subsequent level $s = 1, \ldots, s_{\max}$. Each band is then spatially divided into $k \times k$ non-overlapping patches, resulting in k^2 tokens per band. Each patch (a small tensor of values) is flattened and linearly projected into a fixed-size embedding vector—512 dimensions in our implementation—yielding a uniform token sequence regardless of original band size. This complete sequence of vectors is fed to the DiT at each stage of generation, either all at once (baseline K-Flow) or progressively level by level.

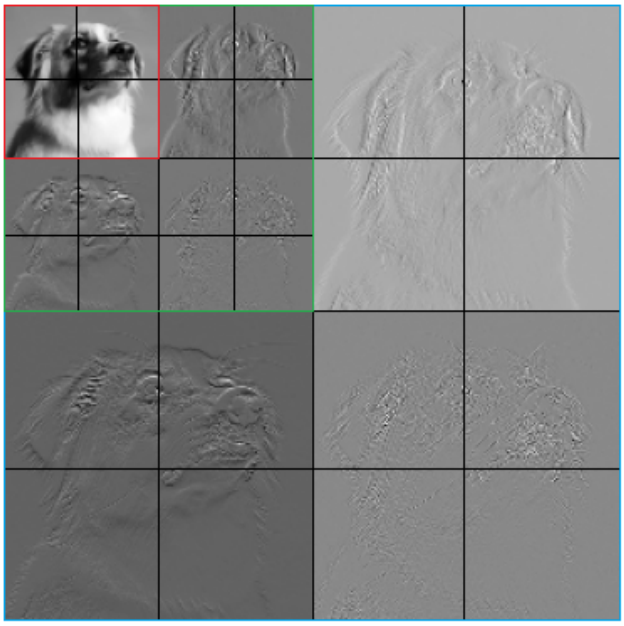

Figure 28: Multi-level wavelet decomposition with patchified token layout. The LL band at level 0 (red box) forms the base representation. Each subsequent level adds high-frequency bands (green for level 1, blue for level 2), with spatial grids indicating how each band is split into tokens.

From there, 2 approaches are possible :

Full Tokenization: In this first approach, the entire multi-level wavelet tensor is tokenized at once. The model is fed the complete sequence of tokens, which includes the low-frequency LL band as well as all high-frequency (HF) bands up to the highest resolution. This means the model must process tokens corresponding to fine-detail bands that are pure noise and are not the focus of the current generation step, even at early stages when we are only filling the LL band.

Progressive (or Selective) Tokenization: This second, more efficient approach is inspired by our modular pipeline. Instead of feeding all tokens at once, we only provide the model with the tokens relevant to the current stage of generation. At a given level s, the input token sequence would consist only of the low-frequency LL band and the high-frequency bands up to level s. This prevents the model from wasting computation on high-frequency noise tokens that are not yet being generated.

The following theoretical analysis directly compares the computational cost, measured in total token-evaluations, of these two proposed wavelet-space approaches. It aims to quantify the significant efficiency benefits of the progressive tokenization strategy.

The full token sequence at scale $s_{\max}$ includes:

- 1 low-frequency band (LL_0) at level 0
- 3 high-frequency bands (LH, HL, HH) for each level $s = 1, \ldots, s_{\max}$

Since each band contributes k^2 patch tokens, the total number of tokens is:

$$N_{\text{tokens}}^{\text{total}} = k^2 \left(1 + 3s_{\max}\right) \tag{33}$$

In contrast, the progressive variant decouples the levels and performs generation explicitly stage by stage. At each scale s, we apply a dedicated portion of the total

function evaluations (NFE) to refine only the new high-frequency bands introduced at that level, while keeping coarser components fixed. This design avoids feeding large regions of pure noise to the model—regions that correspond to wavelet bands not yet meant to be generated—thereby improving efficiency and reducing interference. By restricting attention to the bands currently being filled, we simplify the model's task and reduce the number of irrelevant tokens, which would otherwise consume compute without contributing useful gradients.

Below, we provide a theoretical comparison of token usage and total function evaluations between those two multi-scale adaptations of K-Flow.

Token and Compute Budget Analysis

Throughout, let k^2 be the number of *tokens per wavelet band* (e.g., $8^2 = 64$ when `K_PATCH` $= 8$), and let $s_{\max}$ denote the number of recursive wavelet decompositions applied to the image.

1. Tokens present at a given scale

Single–pass The K-Flow model generates all wavelet bands up to scale $s_{\max}$ in a single denoising trajectory. The token sequence therefore includes:

- one LL band at level 0
- three HF bands (LH, HL, HH) per level from $s = 1$ to $s_{\max}$

Each band is split into k^2 patches, so the total number of tokens is:

$$N_{\text{tok}}^{\text{single}} = k^2 \left(1 + 3s_{\max}\right) \tag{34}$$

Progressive sampler The progressive variant generates bands level by level. At each stage $s \in \{0, \ldots, s_{\max}\}$, we only input the bands needed to generate that level:

$$N_{\text{tok}}^{\text{prog}}(s) = k^2 \left(1 + 3s\right) \tag{35}$$

This includes the LL_0 band and all high-frequency bands from levels 1 to s. As s increases, the number of active tokens grows. Only at the final stage $s = s_{\max}$ do we reach the full token count in (34). However, since each stage solves a smaller ODE with fewer tokens, the overall compute can be reduced (see next section).

2. Total token–evaluations per generated image

Let NFE be the number of ODE function evaluations allocated *in total*.

Paper (single ODE) One trajectory over $t \in [0, s_{\max}]$ on the *full* token sequence:

$$\boxed{T_{\text{single}} = \text{NFE} \cdot k^2 \big(1 + 3s_{\max}\big)} \tag{36}$$

Progressive ($s_{\max} + 1$ ODEs) We split the same NFE evenly across the $s_{\max} + 1$ stages, i.e. $\text{NFE}\,/(s_{\max} + 1)$ evaluations per stage. Summing (35) over all levels gives:

$$\begin{aligned} T_{\text{prog}} &= \sum_{s=0}^{s_{\max}} \frac{\text{NFE}}{s_{\max} + 1} \cdot k^2 \big(1 + 3s\big) \\ &= \frac{\text{NFE} \cdot k^2}{s_{\max} + 1} \left[(s_{\max} + 1) + 3 \tfrac{s_{\max}(s_{\max}+1)}{2} \right] \\ &= \boxed{\text{NFE} \cdot k^2 \left(1 + \tfrac{3}{2} s_{\max}\right)} \end{aligned} \tag{37}$$

3. Efficiency ratio

The fraction of compute the progressive sampler needs relative to the single–pass baseline is:

$$R(s_{\max}) = \frac{T_{\text{prog}}}{T_{\text{single}}} = \frac{1 + \frac{3}{2}s_{\max}}{1 + 3s_{\max}} = \frac{1}{2} + \frac{1}{2\big(1 + 3s_{\max}\big)} \tag{38}$$

This approaches 50% as $s_{\max}$ increases.

Implementation Note. At the time of this work, no official codebase or pre-trained models for K-Flow were released by the original authors. As a result, we reimplemented the method based on the descriptions provided in the paper, including the patchification of images, tokenization, as well as our own architecture. Although the core components of the architecture and training logic are now implemented, we were not yet able to complete a full training run at the time of writing. This remains an active direction for the final weeks of the internship and provides a promising foundation for future experimentation alongside the proposed Wavelet-FM pipeline.

Summary

This section detailed the design and evaluation of our Wavelet-FM architecture, a modular pipeline that separates low- and high-frequency generation. Our results confirmed that this hierarchical approach enables efficient and scalable synthesis and offers significant control over the process. However, our analysis also identified key challenges: the quality of the base low-frequency component remains a primary bottleneck, and the recursive application of the HF-generator can lead to an accumulation of errors at higher resolutions. Additionally, this section provided a theoretical analysis of token efficiency in K-Flow-style architectures, proposing a progressive wavelet-space approach that could yield significant computational savings while being truly "multi-scale" and not simply "coarse-to-fine". Furthermore, a key finding was that high-frequency components themselves are not uniform in complexity; those at lower scales, which carry significant semantic information like edges and contours, proved more challenging to model than the fine-grained local textures at higher resolutions.

5 Conclusion

5.1 Contributions

In this work, we proposed a modular, multi-scale generative modeling architecture that combines Flow Matching with wavelet decomposition. By separating the generation of low-frequency (LL) and high-frequency (HF) components, we enable efficient, scalable image synthesis while preserving semantic coherence and perceptual detail.

5.2 Limitations and discussions

Despite these strengths, our pipeline faces limitations. The quality of the LL component remains a central bottleneck for the final image fidelity. Moreover, the first stage of high-frequency generation, responsible for reconstructing coarse textures at low resolutions, is particularly challenging, as it must recover a semantically meaningful structure from minimal input. Errors introduced at this level are difficult to correct and tend to propagate across subsequent resolution stages.

Finally, while FID provides a useful quantitative benchmark, it does not always align with perceptual quality, especially in the presence of subtle texture artifacts or oversharpening effects, a factor to consider in future evaluations.

5.3 Future Work

Future work could focus on improving the generation of the low-frequency (LL) component. This could involve adapting advanced techniques originally developed for diffusion models—such as improved noise schedules or classifier-free guidance to the Flow Matching framework.

Another promising direction is the integration of semantic or textual conditioning, which would enable controllable image generation across multiple scales, aligning the generation process with user-defined content or style.

Finally, since the main bottlenecks of our pipeline lie in generating coherent low-scale HF components and high-quality LL images, a latent-space approach could be explored. Inspired by latent diffusion, one could train a VAE to encode images into a compact latent representation. The Flow Matching model would then generate samples directly in this latent space, which are subsequently decoded into medium-resolution images (e.g., 128×128). These decoded LL images would then serve as input for the second model responsible for generating the corresponding HF components at higher resolutions (e.g., 256×256 and beyond).

5.4 Broader Impact

More broadly, our results suggest that hierarchical and modular generative models represent a promising direction, especially in contexts where flexibility, resolution scalability, or domain adaptation are important. By decoupling structure and detail, the Wavelet-FM framework offers a simple yet effective approach to multi-resolution image synthesis, with potential for further refinement and application in future work.

An additional direction involves extending the analysis of token efficiency in K-Flow-style DiT architectures. Our preliminary investigation suggests that selective token feeding can significantly reduce inference cost, although empirical validation remains for future work.

Bibliography

Jyoti Aneja, Alexander Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors, 2021. URL https://arxiv.org/abs/2010.02917.

David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbott, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation, 2023. URL https://arxiv.org/abs/2303.04248.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space, 2023. URL https://arxiv.org/abs/2307.08698.

Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G. Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions, 2022. URL https://arxiv.org/abs/2209.05442.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Weitao Du, Shuning Chang, Jiasheng Tang, Yu Rong, Fan Wang, and Shengchao Liu. Flow along the k-amplitude for generative modeling, 2025. URL https://arxiv.org/abs/2504.19353.

Zhengyang Geng, Ashwini Pokle, and J. Zico Kolter. One-step diffusion distillation via deep equilibrium models, 2023. URL https://arxiv.org/abs/2401.08639.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. Wavelet score-based generative modeling, 2022. URL https://arxiv.org/abs/2208.05003.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. URL https://arxiv.org/abs/2206.00364.

Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation, 2022. URL https://arxiv.org/abs/2106.05527.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild, 2015. URL https://arxiv.org/abs/1411.7766.

Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, Inc., USA, 3rd edition, 2008. ISBN 0123743702.

Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models, 2023. URL https://arxiv.org/abs/2301.11706.

Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport, 2021. URL https://arxiv.org/abs/2006.00104.

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Hao Phung, Quan Dao, and Anh Tran. Wavelet diffusion models are fast and scalable image generators, 2023. URL https://arxiv.org/abs/2211.16152.

Prasanna Reddy Pulakurthi, Mahsa Mozaffari, Sohail A. Dianat, Jamison Heard, Raghuveer M. Rao, and Majid Rabbani. Enhancing gans with mmd neural architecture search, pmish activation function, and adaptive rank decomposition. *IEEE Access*, 12:174222–174244, 2024. doi: 10.1109/ACCESS.2024.3485557.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL https://arxiv.org/abs/2202.00512.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets, 2022. URL https://arxiv.org/abs/2202.00273.

Shiv Shankar and Tomas Geffner. Learning straight flows by learning curved interpolants, 2025. URL https://arxiv.org/abs/2503.20719.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models, 2021a. URL https://arxiv.org/abs/2101.09258.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021b. URL https://arxiv.org/abs/2011.13456.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space, 2021. URL https://arxiv.org/abs/2106.05931.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23:1661–1674, 07 2011. doi: 10.1162/NECO_a_00142.

Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, and Mingyuan Zhou. Patch diffusion: Faster and more data-efficient training of diffusion models, 2023. URL https://arxiv.org/abs/2304.12526.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans, 2022. URL https://arxiv.org/abs/2112.07804.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency, 2024. URL https://arxiv.org/abs/2407.02398.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023.

6 Appendix

6.1 Additional Qualitative Samples and real data PCA

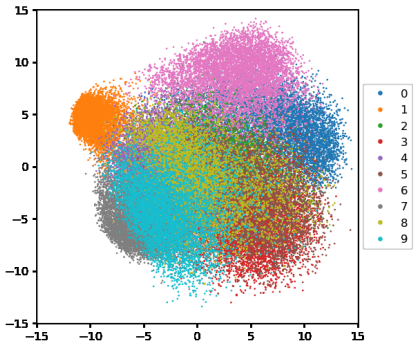

Figure 29: PCA visualization of the latent space: Real structured distribution where digits are well-organized.

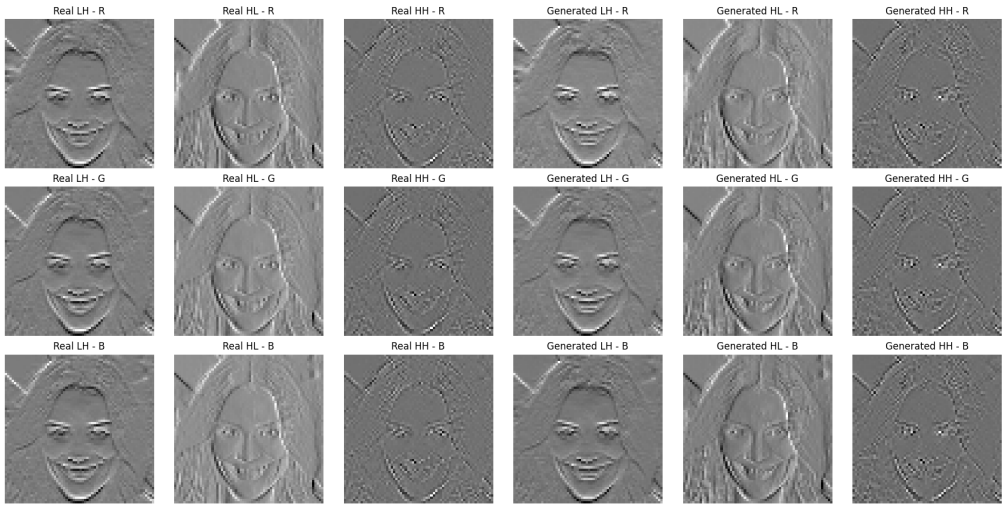

Figure 30: Comparison of real vs. generated high-frequency (HF) wavelet components on CelebA-HQ. Each subfigure shows LH, HL, and HH components across RGB channels.

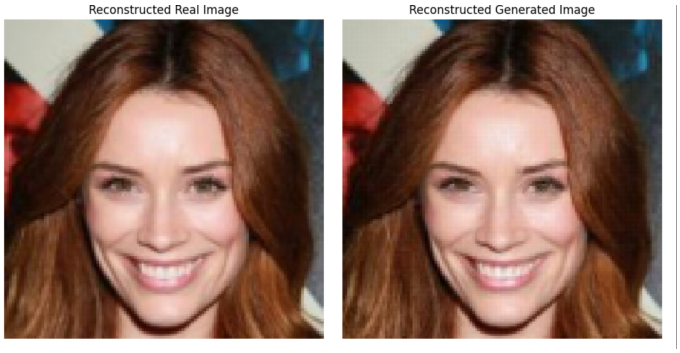

Figure 31: Reconstructed images using real HF components (left) vs generated HF components (right), recombined with the same LL base.

Figure 32: Uncurated LL samples from the model trained on CIFAR-10 at different training steps. Left: step 0 (pure noise); middle: 10k steps; right: 100k steps.

6.2 Training Details

Table 12: Training and architecture hyperparameters for the LL and HF multi-scale models.

Parameter	**LL Model**	**HF Model**
Model	`UNetModelWrapper`	`HFUNet`
Input Channels	3 (RGB)	9 (3 HF bands × 3 channels)
Output Channels	3	12 (9 HF + 3 LL passthrough)
Image Resolution	16×16	16–256 (progressive)
Base Channels	128	128
`channel_mult`	[2, 4]	[1,2,2]
`num_res_blocks`	3	2 (per block)
`num_heads`	6	–
`num_head_channels`	64	–
Attention Resolutions	16, 8	– (uses cross-attention)
Dropout	0.0	0.0
Optimizer	AdamW	AdamW
Learning Rate	1e-4	1e-4
Batch Size	128	32 / 16 / 8 / 4 (adaptive)
Training Steps	200,000	50,000 × 5 resolutions
Gradient Clipping	1.0	1.0
EMA Decay	0.9999	0.9999
Loss	$\|v_\theta - u_t\|^2$ (FM)	$\|v_\theta - u_t\|^2$ (FM)

6.3 Algorithms: FM Loss Approximation and Inference

The following pseudocode summarizes the Monte Carlo approximation used to compute the optimal transport Conditional Flow Matching loss $\mathcal{L}_{\mathrm{CFM}}^{\mathrm{OT}}(\theta)$ during training:

Algorithm 1 Approximation of $\mathcal{L}_{\mathrm{CFM}}^{\mathrm{OT}}(\theta)$

$S \leftarrow 0$
for $i = 0$ to I **do**
  for $j = 0$ to J **do**
    for $k = 0$ to K **do**
      Draw $X_0 \sim p$
      Draw $X_1 \sim q$
      Draw $t \sim \mathcal{U}(0, 1)$
      Compute $X_t = tX_1 + (1 - t)X_0$
      $S \leftarrow S + \|u_t^\theta(X_t) - (X_1 - X_0)\|^2$
    end for
  end for
end for
$S \leftarrow S/(IJK)$
return S

The following algorithm summarizes the inference process to generate samples using a trained Flow Matching model with velocity field u_t^θ.

Algorithm 2 Inference via Flow Matching

Require: Number of steps N, Trained velocity field u_t^θ
Sample $X_0 \sim p_0$ ▷ Initial sample from prior (e.g., Gaussian)
Initialize $\Delta t \leftarrow \frac{1}{N}$
$X \leftarrow X_0$
for $n = 0$ to $N - 1$ **do**
  $t \leftarrow n \cdot \Delta t$
  $X \leftarrow X + \Delta t \cdot u_t^\theta(X)$ ▷ Euler integration step
end for
return X ▷ Generated sample from p_1