Flow Matching for Controlled Image Generation

by

Ahmed Mohammed Ahmed

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Machine Learning and Artificial Intelligence
in the Faculty of Science at Stellenbosch University

Supervisor: Dr Shane Josias

December 2025

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

LLM and AI use acknowledgement

I acknowledge that artificial intelligence (AI) tools, including large language models (LLMs) such as ChatGPT and Grammarly, were used only to assist with understanding theoretical concepts from the assigned readings, LaTeX formatting, grammar checking, code debugging, and formatting citations and references. These tools were not used to write the report or produce any of the research results.

Date: December 2025 .....

Abstract

Continuous normalizing flows [1] trained through flow matching have achieved state-of-the-art results in generative modelling. However, it remains unclear how guidance mechanisms originally developed for diffusion models perform when adapted for flow matching. These mechanisms provide controllable generation by learning conditional distributions and offer flexible control over the trade-off between sample fidelity and diversity.

This work investigates how different conditioning strategies can be incorporated into flow matching to balance this trade-off. Three approaches are investigated: direct label conditioning, classifier guidance, and classifier-free guidance. Each is evaluated across three datasets in order of increasing complexity—MNIST, SVHN, and CIFAR-10—to assess its influence on sample quality and diversity.

The study finds that direct label conditioning enhances sample quality relative to an unconditional model, with the largest gains observed on simpler datasets, MNIST and SVHN, though it does not provide a mechanism for controlling the quality–diversity trade-off. Classifier guidance does not yield any improvement in sample quality compared to label conditioning, possibly because training the classifier on noisy interpolations is difficult. Classifier guidance additionally exhibits behaviour contrary to diffusion models: sample quality and diversity decline as guidance strength increases. Another interesting difference to the diffusion model literature is that optimal guidance scales are much lower for flow models. In contrast, classifier-free guidance showed the expected quality-diversity tradeoff across all datasets. As guidance strength increased, sample quality improved while diversity decreased. Its success may be due to joint training on the generative process itself, rather than reliance on an external classifier, which can be difficult to train. These findings suggest that classifier-free guidance provides the most reliable conditioning mechanism for flow matching.

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisor, Dr Shane Josias. His encouragement, suggestions, mentorship, and guidance have been immensely appreciated. I am also deeply thankful to the Google DeepMind scholarship program for financial support and to Stellenbosch University, with its wonderful lectures, which provided me with an outstanding study environment that facilitated the production of this research. My final appreciation goes to all my family for their understanding and encouragement.

Contents

Declaration i

Abstract ii

1. Introduction 1

2. Mathematical Foundation 5

2.1. The Flow ODE . . . 5

2.2. Flow Matching Formulation . . . 5

2.2.1. Training an Unconditional Flow Model . . . 7

2.2.2. Conditional Generation . . . 7

2.3. Classifier Guidance . . . 8

2.3.1. Deriving Guidance from Bayes' Rule . . . 8

2.3.2. The Guidance Scale Parameter . . . 9

2.3.3. The Classifier . . . 10

2.4. Classifier-Free Guidance . . . 12

2.4.1. Training with Label Dropout . . . 13

3. Methodology 15

3.1. Problem Definition . . . 15

3.2. Experimental Setup . . . 15

3.2.1. Vector Field Parameterisation . . . 15

3.2.2. Classifier Parameterization . . . 16

3.2.3. Datasets . . . 17

3.2.4. Evaluation Metrics . . . 17

3.2.5. Training Hyperparameters . . . 18

3.2.6. Sampling Hyperparameters . . . 18

3.3. Experiments . . . 18

3.3.1. Research Question 1: Effect of Label Conditioning . . . 18

3.3.2. Research Question 2: Impact of Classifier Guidance . . . 19

3.3.3. Research Question 3: Effectiveness of Classifier-Free Guidance . . . 19

3.3.4. Research Question 4: Cross-Dataset Comparison . . . 20

3.3.5. Additional Investigation: Understanding Guidance Failure . . . 20

4. Results **21**
4.1. Research Question 1: Effect of Label Conditioning 21
4.2. Research Question 2: Impact of Classifier Guidance 22
4.3. Research Question 3: Impact of Classifier-Free Guidance 23
4.4. Research Question 4: Cross-Dataset Comparison 24
4.5. Discussion on Guidance Failure 26

5. Conclusion **28**
5.1. Limitations and Future Work 29

Bibliography **30**

Chapter 1

Introduction

Generative models learn a probability distribution $p(\boldsymbol{x})$ from data samples to generate new instances that resemble the training data. The probability distribution is a function that provides the likelihood of a datapoint $\boldsymbol{x} \in \mathbb{R}^d$. For images, $\boldsymbol{x}$ is a vector where each dimension is a pixel value. In this case, $p(\boldsymbol{x})$ provides the likelihood for that specific arrangement of pixels. Consider a simple example where $\boldsymbol{x} \in \mathbb{R}^2$. A Gaussian distribution can model probabilities over the data:

$$p(\boldsymbol{x}) = \frac{1}{2\pi|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{1.1}$$

where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the covariance matrix. This creates an elliptical cloud in space, as shown in Figure 1.1. Points near the mean have a high likelihood, while points far away have a low likelihood. To estimate the distribution $p(\boldsymbol{x})$, a parameterised model $p_\theta(\boldsymbol{x})$ is specified and its parameters θ are learned directly from data by maximising the log-likelihood:

$$\max_\theta \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log p_\theta(\boldsymbol{x})\right]. \tag{1.2}$$

The process of learning $p_\theta(\boldsymbol{x})$ is called density estimation, and has several practical applications. The density allows for evaluating how likely a new data point is, and is useful for anomaly detection [2,3]. It also enables sampling, which is the process of generating new data points that resemble a training dataset. Compression becomes possible as high-probability regions can be encoded with fewer bits [4, 5]. Furthermore, missing values from the data can be estimated based on the underlying learned data distribution [6, 7].

In unconditional generation, new samples are drawn directly from $p(\boldsymbol{x})$. However, unconditional generation provides no control over the generated outputs and desired attributes or characteristics of the samples cannot be specified. This limitation motivates conditional generative modeling, where the model represents $p_\theta(\boldsymbol{x}|y)$, with y representing conditioning information such as class labels, text descriptions, or partial observations. The conditional objective becomes:

$$\max_\theta \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}}\left[\log p_\theta(\boldsymbol{x}|y)\right]. \tag{1.3}$$

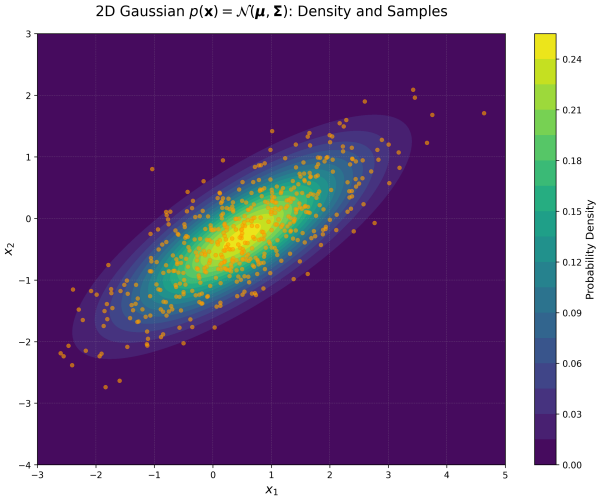

Figure 1.1: A 2D Gaussian distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ showing the probability density, along with generated samples. The density forms an elliptical pattern with the highest likelihood (yellow) at the mean and decreasing likelihood (blue) as distance from the mean increases. Points represent samples drawn from this distribution (orange).

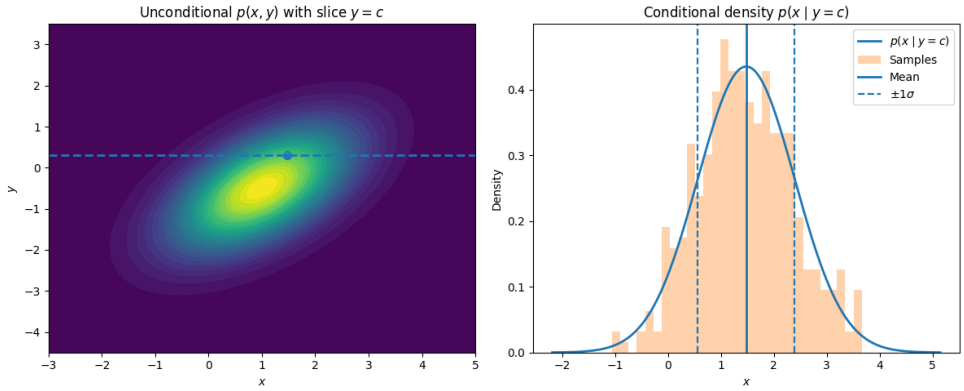

Figure 1.2: The joint distribution $p(\boldsymbol{x}, y)$ with a slice at the value of the conditioned variable $y = c$ showing where conditioning occurs (left). The conditional distribution $p(\boldsymbol{x} \mid y = c)$ is a 1-dimensional Gaussian (right). The histogram shows samples from the conditional distribution, while the blue curve shows the theoretical density.

Figure 1.2 illustrates conditional distributions using a simple 2D example. Conditioning provides several benefits. First, it enables explicit control over generated outputs, allowing specification of desired attributes. Second, conditioning often improves sample quality by constraining the generation process to specific modes of the data distribution. Third, many practical applications require conditional generation, from image inpainting and super-resolution to other image-to-image translation tasks [8].

Diffusion models have recently gained prominence across a wide range of generative tasks [9,10]. In these models, guidance refers to the mechanism used for conditioning [10]. Guidance enables control over the trade-off between diversity and sample quality, allowing the model to balance between generating realistic samples (quality) and maintaining broad coverage of the data distribution (diversity).

This trade-off is not unique to diffusion models and has been achieved in other generative frameworks through different mechanisms. Autoregressive models use temperature scaling [11, 12], where lower temperatures make the model more confident, giving higher quality but less variety, while higher temperatures give more variety but lower quality samples. GANs use the truncation trick [13, 14], sampling only from the center of the latent space where quality is higher, and avoiding the edges where samples are lower quality. This improves quality but reduces diversity. Overall, the quality/diversity tradeoff is important for learning models of high-resolution image data. For diffusion models and continuous normalising flows (or flow models) [15], this trade-off is achieved through conditioning or guidance.

Continuous normalising flows trained through flow matching [1] provide an alternative to diffusion models and have recently achieved state-of-the-art results in image and video generation. A flow model is a generative model that learns a vector field that transforms samples from a simple base distribution (such as Gaussian noise) into the target data distribution. Unlike diffusion models that gradually add and remove noise through a fixed schedule, flow matching directly learns the optimal transport path between distributions.

Flow matching and diffusion models share the common goal of learning time-dependent transformations; however, training flow models can be simpler. Score matching is unnecessary—instead, the vector field is trained via direct regression using linear interpolation between noise and data. This naturally raises an important question: if guidance strategies that have proven effective in diffusion models are applied to flow matching, do they yield similar benefits?

To investigate this, three forms of conditioning are explored in flow models: (1) basic label conditioning, where the vector field receives label information during training; (2) classifier guidance, which modifies the vector field during sampling using a pretrained classifier; and (3) classifier-free guidance, which achieves similar control without relying on an external classifier. Together, these approaches aim to evaluate how guidance can influence controllability and sample quality in flow-matching models.

Four research questions are posed to evaluate the impact of conditioning and guidance in flow models:

1. How does label conditioning influence sample quality compared to unconditional generation?

2. How does classifier guidance affect sample quality across different guidance scales?

3. How do classifier guidance and classifier-free guidance compare in terms of their effect on sample quality and diversity?

4. How do different guidance mechanisms behave across datasets of varying complexity?

The remainder of the report is structured as follows. The mathematical foundations of flow matching are presented in Chapter 2, covering the flow ODE formulation, conditional generation, and guidance mechanisms. Chapter 3 describes the experimental methodology for exploring the abovementioned research questions. This chapter includes details on datasets, model architectures, and evaluation metrics. Chapter 4 presents the experimental results and analysis. Finally, Chapter 5 concludes with a summary of findings and directions for future work.

Chapter 2

Mathematical Foundation

2.1. The Flow ODE

A continuous normalising flow is defined by an ordinary differential equation (ODE), which describes conditions on a time-dependent trajectory, $\boldsymbol{x}_t$. A trajectory starts at a specific point called the initial condition. The trajectory follows a vector field that describes the direction. The vector field $\boldsymbol{u}_t(\boldsymbol{x}_t)$ defines the velocity at position $\boldsymbol{x}_t$ and time t, where $\boldsymbol{x}_t \in \mathbb{R}^d$ represents the state at time $t \in [0, 1]$:

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{u}_t(\boldsymbol{x}_t), \quad \boldsymbol{x}_0 \sim p_{\text{init}}, \quad \boldsymbol{x}_1 \sim p_{\text{data}}. \tag{2.1}$$

The flow is a collection of trajectories that follow the ODE, over all possible initial conditions. A vector field defines an ODE, a trajectory is a solution to this ODE, and a flow is a collection of trajectories for various initial conditions [1].

The vector field $\boldsymbol{u}_\theta$ is usually parameterised with a neural network with parameters θ. Once the parameters are learned, new samples can be generated by solving the probability flow ODE in equation (2.1) starting from $\boldsymbol{x}_0$ as an initial condition sampled from a random noise distribution $p_0(\boldsymbol{x})$. Equation (2.1) can be solved numerically from $t = 0$ to $t = 1$ by using any standard ODE solver for the following integral equation:

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \int_0^1 \boldsymbol{u}_\theta(\boldsymbol{x}_t, t)\, dt. \tag{2.2}$$

2.2. Flow Matching Formulation

The flow matching framework aims to learn the parameterised $\boldsymbol{u}_\theta$ by regressing against a marginal vector field $\boldsymbol{u}_t(\boldsymbol{x})$ that defines a marginal probability path $p_t(\boldsymbol{x})$ connecting a source distribution p_0 (such as Gaussian noise) to a target data distribution p_1 [1].

The marginal flow matching objective is given by:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, \boldsymbol{x}\sim p_t} \left\| \boldsymbol{u}_\theta(\boldsymbol{x}, t) - \boldsymbol{u}_t(\boldsymbol{x}) \right\|_2^2. \tag{2.3}$$

However, computing the marginal probability path and vector field is intractable. Conditional flow matching addresses this by introducing conditional probability paths and vector fields.

The conditional vector field can be defined by first defining a linear interpolation between a sample $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x})$ and a sample $\boldsymbol{x}_1 \sim p_1(\boldsymbol{x})$, with $t \sim \mathcal{U}(0, 1)$:

$$\boldsymbol{x}_t = (1 - t)\boldsymbol{x}_0 + t\boldsymbol{x}_1. \tag{2.4}$$

The corresponding ground truth conditional vector field for this path becomes:

$$\boldsymbol{u}_t(\boldsymbol{x}_t|\boldsymbol{x}_1) = \boldsymbol{x}_1 - \boldsymbol{x}_0. \tag{2.5}$$

The goal is to train a neural network, $\boldsymbol{u}_\theta(\boldsymbol{x}, t)$, to approximate this vector field. The conditional flow matching training objective is mean squared error loss between the $\boldsymbol{u}_\theta(\boldsymbol{x}, t)$ and the target conditional vector field, minimised over samples from the data distribution:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,p_0,p_1}[\|\boldsymbol{u}_\theta(\boldsymbol{x}_t, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2]. \tag{2.6}$$

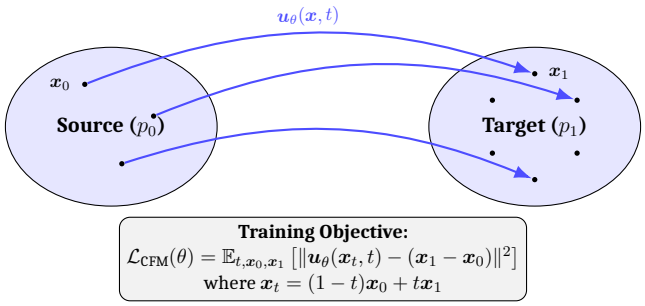

Figure 2.1: The flow matching framework provides a way to learn a neural network vector field that transforms samples from a source distribution p_0 to a target distribution p_1.

The target vector field in Equation (2.6) provides the direction of optimal transport between the source and target distributions. Figure 2.1 illustrates the flow matching framework.

In summary, flow matching defines a continuous transformation from a simple base distribution to the data distribution using an ordinary differential equation (ODE). The ODE specifies a trajectory $\boldsymbol{x}_t$ that evolves smoothly from time $t = 0$ to $t = 1$, mapping the base samples to data samples. At each point along this trajectory, the conditional vector field $\boldsymbol{u}_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$ determines the instantaneous direction of transformation [1].

2.2.1. Training an Unconditional Flow Model

As mentioned earlier, the vector field $\boldsymbol{u}_\theta(\boldsymbol{x}_t, t)$ is parameterised by a neural network with learnable parameters θ. The network is trained to approximate the conditional vector field defined in Equation (2.5) by minimising the mean squared error between predicted and true velocities. The training procedure is shown in Algorithm 2.1.

Algorithm 2.1: Training Unconditional Flow Model

Input: Training dataset $\mathcal{D}$, epochs E, learning rate η
Output: Trained model parameters θ
Initialize network parameters θ randomly
Initialize optimizer (Adam) with learning rate η
for epoch $= 1$ to E **do**
  for each batch $(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_1^{(B)})$ from $\mathcal{D}$ **do**
    Sample noise $\boldsymbol{x}_0^{(i)} \sim \mathcal{N}(\mathbf{0}, I)$ for $i = 1, \ldots, B$
    Sample times $t \sim \mathcal{U}(0, 1)$ for $i = 1, \ldots, B$
    for $i = 1$ to B **do**
      $\boldsymbol{x}_t^{(i)} = (1 - t^{(i)})\boldsymbol{x}_0^{(i)} + t^{(i)}\boldsymbol{x}_1^{(i)}$
      $\boldsymbol{u}_{\text{true}}^{(i)} = \boldsymbol{x}_1^{(i)} - \boldsymbol{x}_0^{(i)}$
      $\boldsymbol{u}_{\text{pred}}^{(i)} = \boldsymbol{u}_\theta(\boldsymbol{x}_t^{(i)}, t^{(i)})$
      $\ell^{(i)} = \|\boldsymbol{u}_{\text{pred}}^{(i)} - \boldsymbol{u}_{\text{true}}^{(i)}\|^2$
    end for
    $\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B} \ell^{(i)}$
    $\boldsymbol{g} = \nabla_\theta \mathcal{L}$
    $\boldsymbol{g} = \text{clip}(\boldsymbol{g}, \text{max\_norm} = 1.0)$
    $\theta = \text{Adam}(\theta, \boldsymbol{g}, \eta)$
  end for
end for
return θ

The training process follows standard supervised regression. Noise and data are sampled independently for each batch, linear interpolation paths are constructed, target velocities are computed, and the network is trained to match these targets through mean squared error minimisation. Once trained, the model can generate new samples by solving the ODE in Equation (2.2) starting from noise.

2.2.2. Conditional Generation

The flow matching formulation described above leads to an unconditional generative model that estimates $p(\boldsymbol{x})$ without any label information. The vector field $\boldsymbol{u}_\theta(\boldsymbol{x}_t, t)$ depends only on the current state and time. However, many practical applications require conditional generation, where specific attributes of generated samples can be controlled [10].

Flow matching can be adapted for conditional generation by introducing conditioning information y during training and sampling. The conditioning variable y may represent class labels, text

descriptions, or other attributes that guide the generation process. This modification changes three key aspects of the flow matching framework: the vector field structure, the training process, and the objective function.

The vector field is modified to accept the conditioning variable as an additional input. Instead of $\boldsymbol{u}_\theta(\boldsymbol{x}_t, t)$, the conditional vector field becomes $\boldsymbol{u}_\theta(\boldsymbol{x}_t, t, y)$. The training process is adapted to use labeled data pairs. For labeled data pairs $(\boldsymbol{x}_1, y) \sim p_{\text{data}}(\boldsymbol{x}_1, y)$, the conditional flow matching objective is defined as [1]:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{(\boldsymbol{x}_1,y)\sim p_{\text{data}},\, t\sim\mathcal{U}(0,1),\, \boldsymbol{x}\sim p_t(\cdot\,|\,\boldsymbol{x}_1)} \Big\| \boldsymbol{u}_\theta(\boldsymbol{x}, t, y) - \boldsymbol{u}_t(\boldsymbol{x}\,|\,\boldsymbol{x}_1) \Big\|_2^2. \tag{2.7}$$

The objective function incorporates label information through the sampling process. Still considering a linear interpolation between source and target samples, the conditional flow matching objective is:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{(\boldsymbol{x}_1,y),\, \boldsymbol{x}_0,\, t} \Big\| \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, y) - (\boldsymbol{x}_1 - \boldsymbol{x}_0) \Big\|_2^2. \tag{2.8}$$

The target vector field $\boldsymbol{u}_t(\boldsymbol{x}\,|\,\boldsymbol{x}_1) = \boldsymbol{x}_1 - \boldsymbol{x}_0$ remains unchanged from the unconditional case. The conditioning affects only how the neural network processes its input. During sampling, different values of y guide the generation process toward samples with desired attributes, enabling the model to learn the conditional distribution $p(\boldsymbol{x}|y)$ instead of the marginal distribution $p(\boldsymbol{x})$.

2.3. Classifier Guidance

Having established the framework for conditional and unconditional generation through flow matching in Section 2.2.2, this section presents the mathematical derivations for classifier guidance mechanisms. Understanding these derivations provides the necessary foundation before examining their practical implementation and experimental evaluation in subsequent Sections.

2.3.1. Deriving Guidance from Bayes' Rule

The aim is to generate samples from the conditional distribution $p(\boldsymbol{x} \mid y)$, where y is a class label. Starting with Bayes' rule:

$$p(\boldsymbol{x} \mid y) = \frac{p(\boldsymbol{x}, y)}{p(y)}, \tag{2.9}$$

where the numerator $p(\boldsymbol{x}, y)$ is the joint probability of $\boldsymbol{x}$ and y occurring together, and the denominator $p(y)$ is the marginal probability of y.

The joint probability can be rewritten as:

$$p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y \mid \boldsymbol{x}). \tag{2.10}$$

Substituting Equation 2.10 into Equation 2.9 yields:

$$p(\boldsymbol{x} \mid y) = \frac{p(\boldsymbol{x})p(y \mid \boldsymbol{x})}{p(y)}. \tag{2.11}$$

Taking the logarithm of both sides gives:

$$\log p(\boldsymbol{x} \mid y) = \log p(\boldsymbol{x}) + \log p(y \mid \boldsymbol{x}) - \log p(y). \tag{2.12}$$

Taking the gradient with respect to $\boldsymbol{x}$ yields:

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid y) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}). \tag{2.13}$$

Equation 2.13 is the fundamental equation for guidance [10]. The left-hand side represents the score of the conditional distribution, indicating directions in $\boldsymbol{x}$-space that lead toward samples of a particular class y. The right-hand side decomposes this into two terms. The first term, $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$, is the unconditional score, which determines how probability density varies without any conditioning information. The second term, $\nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x})$, is the classifier gradient. Here $p(y \mid \boldsymbol{x})$ represents a classifier trained to predict class y from sample $\boldsymbol{x}$. The gradient $\nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x})$ indicates directions in $\boldsymbol{x}$-space where the classifier's confidence in predicting class y increases.

2.3.2. The Guidance Scale Parameter

It is possible to weight the classifier term from Equation 2.13 with a scalar variable w, known as the guidance scale. Using a guidance scale in this way leads to sampling from a modified distribution given by:

$$p_w(\boldsymbol{x} \mid y) \propto p(\boldsymbol{x})p(y \mid \boldsymbol{x})^w. \tag{2.14}$$

By modifying Equation 2.13 with the guidance scale, the following is obtained:

$$\nabla_{\boldsymbol{x}} \log p_w(\boldsymbol{x} \mid y) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + w\nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}). \tag{2.15}$$

Integrating both sides with respect to $\boldsymbol{x}$ gives:

$$\log p_w(\boldsymbol{x} \mid y) = \log p(\boldsymbol{x}) + w \log p(y \mid \boldsymbol{x}) + \text{const}. \tag{2.16}$$

Taking the exponential yields:

$$p_w(\boldsymbol{x} \mid y) = Cp(\boldsymbol{x})p(y \mid \boldsymbol{x})^w, \tag{2.17}$$

where C is a normalization constant. When $w = 0$, this reduces to:

$$p_0(\boldsymbol{x} \mid y) \propto p(\boldsymbol{x}), \tag{2.18}$$

which corresponds to unconditional generation. When $w = 1$, the distribution becomes:

$$p_1(\boldsymbol{x} \mid y) \propto p(\boldsymbol{x})p(y \mid \boldsymbol{x}). \tag{2.19}$$

Since $p(\boldsymbol{x})$ is constant for all values of y, Equation 2.19 provides the conditional distribution $p(\boldsymbol{x} \mid y)$, up to normalization. When $w > 1$, the classifier probability is raised to a power greater than one, and amplifies the differences between high and low probability regions. This occurs because raising a probability greater than 0.5 to a higher power makes it closer to 1, while raising a probability less than 0.5 to a higher power makes it closer to 0, thereby sharpening the distribution [10]. This amplification allows for stronger conditioning at the cost of sample diversity.

2.3.3. The Classifier

For this section, it is assumed that a flow model has been trained for unconditional generation and that a classifier must be obtained. Having derived the mathematical foundations, how to implement classifier guidance in practice is now described. The classifier $p(y \mid \boldsymbol{x}_t, t)$ is parameterized by a neural network with parameters ϕ. The classifier takes as input noise-interpolated images $\boldsymbol{x}_t$ and time t, and outputs class logits or probabilities for classes y [10].

The classifier architecture (Figure 2.2) consists of three components. First, time is encoded using sinusoidal positional embeddings [16]. For a scalar time value t, the embedding is given by:

$$\boldsymbol{e}_t = [\cos(2\pi f_1 t), \sin(2\pi f_1 t), \ldots, \cos(2\pi f_K t), \sin(2\pi f_K t)], \tag{2.20}$$

where $f_1, \ldots, f_K$ are frequencies chosen logarithmically spaced between low and high frequencies. This vector is passed through a small neural network (learned end-to-end during classifier training) to obtain a time embedding of dimension d_t.

Second, the image is processed through a convolutional neural network to extract features. The network has multiple layers with downsampling to build a hierarchy of features. Residual connections and group normalization [17] are used for stability, where the final output is a feature vector of dimension d_x obtained by global average pooling.

Third, the time embedding and image features are concatenated to obtain a combined vector of dimension d_t+d_x, which is then passed through fully connected layers with dropout (probability 0.1) to produce class logits.

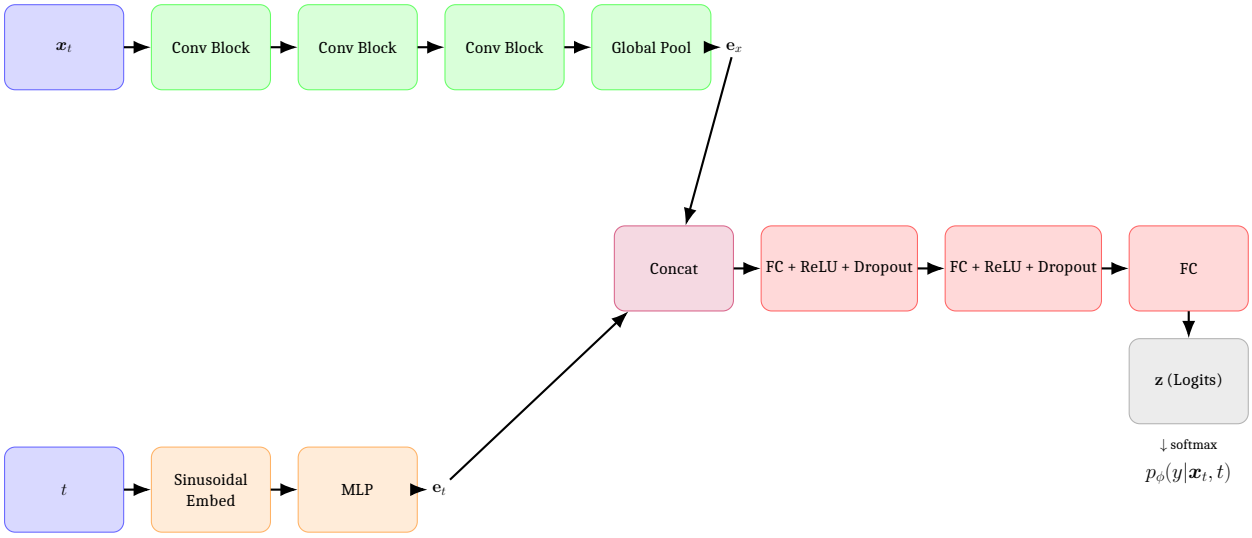

Figure 2.2: Time-conditioned classifier architecture.

For each data sample with label y, a time t is sampled uniformly from $[0, 1]$ and a noisy version is created using the linear interpolation path $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$.The classifier is then trained using softmax and cross-entropy loss to predict the label corresponding to $\boldsymbol{x}_1$ from each image on the interpolation path. The classifier is trained using the procedure shown in Algorithm 2.2.

Algorithm 2.2: Training Time-Conditioned Classifier

Input: Training dataset $\mathcal{D}$ with labels, epochs E, learning rate η
Output: Trained classifier parameters ϕ
Initialize classifier parameters ϕ randomly
Initialize optimizer with learning rate η
for epoch $= 1$ to E **do**
  for each batch $(\boldsymbol{x}_1^{(i)}, y^{(i)})$ for $i = 1, \ldots, B$ from $\mathcal{D}$ **do**
    for $i = 1$ to B **do**
      Sample $\boldsymbol{x}_0^{(i)} \sim \mathcal{N}(\mathbf{0}, I)$ and $t^{(i)} \sim \mathcal{U}(0, 1)$
      $\boldsymbol{x}_t^{(i)} = (1 - t^{(i)})\boldsymbol{x}_0^{(i)} + t^{(i)}\boldsymbol{x}_1^{(i)}$
      $\boldsymbol{e}_t^{(i)} = \text{TimeEmbed}(t^{(i)})$, $\boldsymbol{e}_x^{(i)} = \text{CNN}(\boldsymbol{x}_t^{(i)})$
      $\boldsymbol{e}^{(i)} = [\boldsymbol{e}_t^{(i)}; \boldsymbol{e}_x^{(i)}]$
      $\boldsymbol{z}^{(i)} = \text{MLP}(\boldsymbol{e}^{(i)})$
      $\ell^{(i)} = -\log \frac{\exp(z_{y^{(i)}}^{(i)})}{\sum_k \exp(z_k^{(i)})}$
    end for
    $\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B} \ell^{(i)}$
    Compute gradients and update ϕ
  end for
end for
return ϕ

With both an unconditional flow model and the classifier trained, guided sampling can be performed at test time to generate images of a specific class. The procedure starts from Gaussian noise $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, I)$. The time interval $[0, 1]$ is discretized into N steps (depending on the ODE solver). At each time step t, the unconditional vector field is first computed by evaluating the flow model: $\boldsymbol{u} = \boldsymbol{u}_\theta(\boldsymbol{x}, t)$. The classifier gradient is then computed via automatic differentiation. These are combined according to Equation 2.15, leading to an approximation of the classifier gradient in terms of the conditional and unconditional vector field, to form the guided vector field:

$$\boldsymbol{u}_{\text{guided}}(\boldsymbol{x}, t, y) = \boldsymbol{u}_\theta(\boldsymbol{x}, t) + w\nabla_{\boldsymbol{x}} \log p_\phi(y \mid \boldsymbol{x}, t), \tag{2.21}$$

where w is the guidance scale. The ODE is then integrated using this guided field: $\boldsymbol{x}_{t+\Delta t} = \boldsymbol{x}_t + \Delta t \boldsymbol{u}_{\text{guided}}(\boldsymbol{x}_t, t, y)$. After N steps, the initial noise is transformed into a generated sample. If the flow model and classifier are well-trained and the guidance scale is appropriate, this sample is a realistic image of class y.

2.4. Classifier-Free Guidance

Classifier-free guidance (CFG) [18] achieves guidance without training a separate classifier. Instead, a single conditional flow model is trained that can predict both conditional and unconditional vector fields. This approach offers several advantages: it requires training only one model instead of two, eliminates the need for classifier gradients during sampling, and can produce better results in practice.

From Equation 2.13, the classifier gradient can be expressed as:

$$\nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid y) - \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}). \tag{2.22}$$

Equation 2.22 shows that the classifier gradient equals the difference between the conditional and unconditional scores. In flow matching, the vector field $\boldsymbol{u}_\theta(\boldsymbol{x}, t)$ is related to the score through the continuity equation [19]. Under the probability paths used in flow matching, the score function is approximated by the velocity field, which allows score differences to be expressed as velocity field differences. This leads to an approximation of the classifier gradient in terms of the conditional and unconditional vector field:

$$\nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}, t) \approx \boldsymbol{u}_{\text{cond}}(\boldsymbol{x}, t, y) - \boldsymbol{u}_{\text{uncond}}(\boldsymbol{x}, t), \tag{2.23}$$

where $\boldsymbol{u}_{\text{cond}}$ denotes the conditional vector field for generating samples of class y (predicted by the model when conditioned on a label), and $\boldsymbol{u}_{\text{uncond}}$ denotes the unconditional vector field obtained when the same model is evaluated without class information.

It is important to note that a single model can predict both $\boldsymbol{u}_{\text{cond}}$ and $\boldsymbol{u}_{\text{uncond}}$ through a training technique called label dropout, described in the following section. If such a model is available, guidance can be computed without a separate classifier. Substituting Equation 2.23 into Equation 2.21 yields:

$$\boldsymbol{u}_{\text{guided}}(\boldsymbol{x}, t, y) = \boldsymbol{u}_{\text{uncond}}(\boldsymbol{x}, t) + w[\boldsymbol{u}_{\text{cond}}(\boldsymbol{x}, t, y) - \boldsymbol{u}_{\text{uncond}}(\boldsymbol{x}, t)]. \tag{2.24}$$

Rearranging Equation 2.24 gives:

$$\boldsymbol{u}_{\text{guided}}(\boldsymbol{x}, t, y) = (1 - w)\boldsymbol{u}_{\text{uncond}}(\boldsymbol{x}, t) + w\boldsymbol{u}_{\text{cond}}(\boldsymbol{x}, t, y). \tag{2.25}$$

Equation 2.25 shows that $\boldsymbol{u}_{\text{guided}}(\boldsymbol{x}, t, y)$ is a weighted combination of unconditional and conditional vector fields, controlled by the guidance scale w.

2.4.1. Training with Label Dropout

To train a single model that predicts both conditional and unconditional vector fields, label dropout is employed. During training, some labels are randomly replaced with a special null token that indicates no label is provided. This forces the model to learn both modes of generation simultaneously.

The label space is extended from K classes to $K + 1$ classes, where classes $0, 1, \ldots, K - 1$ are the real classes and class K is the null class representing no conditioning. An embedding table with $K + 1$ entries is created as a learned parameter matrix of size $(K + 1) \times d_{\text{embed}}$, where each row is a learned embedding vector. Each class index is mapped to its corresponding row in this table to obtain the class embedding. The embedding is learned end-to-end during training through backpropagation from conditional flow matching objective.

During training, for each sample with true label y, the label is randomly replaced with the null token with probability p_{drop}. The model architecture accepts a class label as input, retrieves the corresponding embedding from the learned embedding table, and concatenates this class embedding with the time embedding (obtained from sinusoidal functions as in Equation 2.20). These concatenated embeddings are then added to the intermediate feature maps at multiple layers throughout the U-Net architecture using adaptive layer normalization [20]. When the null token is provided, the model learns to predict the unconditional vector field $\boldsymbol{u}_{\text{uncond}}(\boldsymbol{x}, t)$. When a real label is provided, the model learns to predict the conditional vector field $\boldsymbol{u}_{\text{cond}}(\boldsymbol{x}, t, y)$ for that class.

The dropout probability p_{drop} is an important hyperparameter. A typical value is $p_{\text{drop}} = 0.1$, meaning 10% of samples use the null token and 90% use true labels. If p_{drop} is too low (such as 1%), the model does not see enough unconditional examples and unconditional generation is poor. Conversely, if p_{drop} is too high (such as 50%), the model does not see enough labeled examples and conditional generation suffers. Values between 0.1 and 0.2 typically work well in practice. It is important to note that while classifier-free guidance does not require training a separate classifier, it does require labeled data during training of the flow model.

At sampling time, the model is evaluated twice per step: once with the target class label to obtain the conditional vector field $\boldsymbol{u}_{\text{cond}}(\boldsymbol{x}, t, y)$, and once with the null token to obtain the unconditional vector field $\boldsymbol{u}_{\text{uncond}}(\boldsymbol{x}, t)$. These are then combined according to Equation 2.25 to form the guided vector field. This eliminates the need for a separate classifier while retaining the benefits of guidance.

Chapter 3

Methodology

Having established the mathematical foundations of guidance, the specific methodological choices for addressing the research questions are now described.

3.1. Problem Definition

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ are data samples and $y_i \in \{0, 1, \ldots, K-1\}$ are class labels, the aim is to learn a conditional distribution $p(\boldsymbol{x}|y)$ that allows generation of samples from specific classes.

To answer the research questions, four approaches are investigated. First, unconditional generation learns $p(\boldsymbol{x})$ without label information. This serves as a baseline. The second approach, basic label conditioning, uses learned embeddings to condition the vector field on labels. In the third approach, classifier guidance applies gradients from a separate classifier to steer generation toward target classes. Finally, classifier-free guidance trains a single model with label dropout, enabling flexible control at test time.

3.2. Experimental Setup

This section describes the architecture, training procedure, datasets, and evaluation metrics used throughout the experiments.

3.2.1. Vector Field Parameterisation

A U-Net architecture [21] is chosen to parameterize the vector field $\boldsymbol{u}_\theta(\boldsymbol{x}, t)$. Simpler architectures like multilayer perceptrons can be used too. It was found that MLPs struggle with spatial information in images. U-Net preserves spatial structure throughout the network. This makes it more suitable for image generation tasks. The U-Net implementation and hyperparameters are adapted from [10] to achieve the best possible results.

The U-Net implementation includes residual blocks with group normalization [17] for stable training. Self-attention layers [16] at lower resolutions capture long-range dependencies. Time and class embeddings are added through adaptive layer normalization [20]. This lets the network modulate features using conditioning information. Base channels of 128 are used, with channel multipliers of [1, 2, 4, 4]. Attention is applied at resolutions 16 and 8. Two residual blocks appear at each resolution. The embedding dimension is 512 for both time and class embeddings.

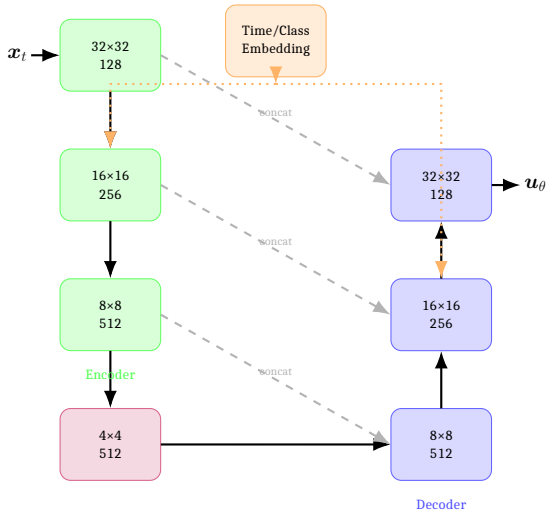

Figure 3.1: U-Net architecture with encoder-decoder structure and skip connections. Numbers indicate spatial resolution and channel dimensions at each level.

3.2.2. Classifier Parameterization

For the classifier guidance experiments, a time-conditioned classifier is trained. The classifier architecture is based on a convolutional neural network. It is similar to the encoder portion of the U-Net above, but outputs class logits rather than a vector field. The architecture includes: (1) sinusoidal time embeddings, (2) convolutional layers with group normalization and residual connections to extract image features, (3) global average pooling to obtain a fixed-size feature vector, and (4) fully connected layers with dropout (probability 0.1) to produce class logits. The classifier is trained for 25 epochs with cross-entropy loss.

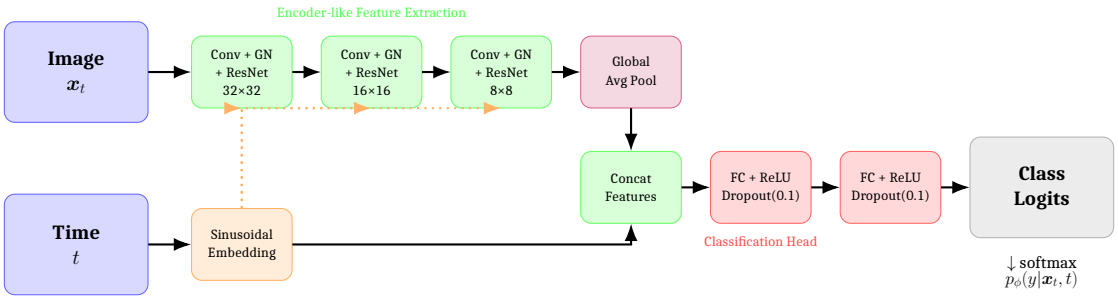

Figure 3.2: Time-conditioned classifier architecture.

3.2.3. Datasets

The methods are evaluated on three datasets: MNIST [22], SVHN [23], and CIFAR-10 [24]. Figure 3.3 shows representative samples from each dataset.

MNIST consists of 60,000 training images of handwritten digits (0-9) at 28×28 grayscale resolution. Its clear class boundaries make it suitable for initial validation.

SVHN contains 73,257 training images of street view house numbers at 32×32 color resolution. This dataset is more challenging than MNIST. The challenges come from complex backgrounds, varying fonts, and different lighting conditions.

CIFAR-10 contains 50,000 training images across 10 natural object classes at 32×32 color resolution. This dataset is the most complex of the three. There is substantial intra-class variation, such as differences in dog breed or viewing angle. There is also inter-class ambiguity.

Figure 3.3: Example training images from the three datasets.

3.2.4. Evaluation Metrics

Three metrics are used to assess the quality of the generated samples.

Fréchet Inception Distance (FID) [25] measures the distance between distributions of real and generated images in feature space. FID is computed as the distance between two Gaussian distributions. These are fitted to Inception-v3 pool3 features (2048-dimensional) extracted from real and generated images:

$$\text{FID} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|^2 + \text{Tr}(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2\sqrt{\boldsymbol{\Sigma}_r \boldsymbol{\Sigma}_g}), \tag{3.1}$$

Here, $\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r$ are the mean and covariance of real data features. $\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g$ are the mean and covariance of generated data features. Lower FID indicates better overall sample quality.

Precision and recall [26] provide information on sample quality and diversity. Precision determines what fraction of generated samples are of high quality (fall within the real data manifold), while recall measures what fraction of the real data distribution is covered by the generated samples. VGG-16 features are extracted from the second fully connected layer, yielding 4096-dimensional vectors. Higher precision indicates better sample quality, while higher recall indicates better coverage of the data distribution.

3.2.5. Training Hyperparameters

For the training, the AdamW optimizer [27] is employed. The learning rate is set to 2×10^{-4} for flow models and 1×10^{-3} for classifiers. A cosine annealing schedule gradually reduces the learning rate over training. The batch size is 128 for MNIST and SVHN, and 64 for CIFAR-10 due to memory constraints. Weight decay is set to 0.01. Gradient clipping at norm 1.0 is applied for stability. An exponential moving average (EMA) of model parameters with a decay of 0.9999 is maintained, and the EMA model is used for evaluation to improve sampling quality.

3.2.6. Sampling Hyperparameters

For sampling, Heun's method is used as the ODE solver with 50 integration steps. All models are evaluated using 10,000 generated samples to ensure comparable metric estimates. For guided generation, equal per-class counts are used (1,000 samples per class for 10-class datasets).

3.3. Experiments

Having described the model parameterisation in the previous section, this section presents the experimental design for each research question.

3.3.1. Research Question 1: Effect of Label Conditioning

To investigate how sample quality changes when label conditioning is used compared to unconditional generation, two models with identical architectures (described in Section 3.2) are trained, differing only in whether label information is provided during training.

The first model is trained without label information, learning only the unconditional distribution $p(\boldsymbol{x})$. The second model is trained with class labels, where labels are encoded as learned embeddings and concatenated with time embeddings throughout the network, learning the conditional distribution $p(\boldsymbol{x}|y)$.

By comparing the unconditional and conditional approaches using FID, precision, and recall, the isolated effect of direct label conditioning on sample quality is quantified. Lower FID indicates better overall sample quality, while precision and recall reveal whether conditioning improves sample fidelity or distribution coverage. This controlled comparison answers whether providing labels during training improves generation quality.

3.3.2. Research Question 2: Impact of Classifier Guidance

To investigate how classifier guidance affects generation quality across different guidance scales, a classifier is trained and then used to provide guidance. During generation, the classifier guides the sampling process by modifying the vector field:

$$\boldsymbol{u}_{\text{guided}}(\boldsymbol{x}_t, t, y) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t) + w \cdot \nabla_{\boldsymbol{x}_t} \log p_\phi(y|\boldsymbol{x}_t, t), \tag{3.2}$$

where w controls the strength of guidance. Guidance scales $w \in \{0.0, 1.0, 2.0, 4.0, 6.0\}$ are tested, ranging from no guidance to stronger guidance.

By varying w and measuring FID, precision, and recall, this experiment determines how guidance strength affects the quality-diversity trade-off. These metrics are computed from a balanced number of samples per class. Comparing these metrics across guidance scales reveals whether classifier guidance successfully trades diversity for quality or introduces artifacts that degrade overall performance. The $w = 0$ baseline compares with unconditional generation from Research Question 1, isolating the effect of balanced class sampling. The optimal guidance scale for classifier guidance can thus be determined.

3.3.3. Research Question 3: Effectiveness of Classifier-Free Guidance

To investigate how classifier-free guidance compares with classifier guidance, a single model is trained to learn both conditional and unconditional distributions simultaneously via label dropout. During training, class labels are randomly replaced with a null token ($\varnothing$) with probability $p_{\text{drop}} = 0.1$, forcing the model to learn both $p(\boldsymbol{x})$ and $p(\boldsymbol{x}|y)$.

At sampling time, both conditional and unconditional vector fields are computed and combined:

$$\boldsymbol{u}_{\text{CFG}}(\boldsymbol{x}_t, t, y) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, \varnothing) + w \cdot (\boldsymbol{u}_\theta(\boldsymbol{x}_t, t, y) - \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, \varnothing)). \tag{3.3}$$

The same guidance scales $w \in \{0.0, 1.0, 2.0, 4.0, 6.0\}$ as classifier guidance is tested. FID, precision, and recall are measured for each scale.

This experimental design enables direct comparison with classifier guidance from Research Question 2, as both methods use identical guidance scales and evaluation procedures. By comparing metrics across guidance scales for both approaches, this experiment reveals which method better balances sample quality and diversity. Additionally, comparing the CFG at $w = 1$ with the conditional model from Research Question 1 isolates whether the label dropout training strategy affects generation quality. This controlled comparison answers whether CFG is more effective than classifier guidance.

3.3.4. Research Question 4: Cross-Dataset Comparison

To examine how guidance mechanisms behave differently across datasets of varying complexity, all four model types are trained on three datasets: MNIST (simple), SVHN (moderate complexity), and CIFAR-10 (high complexity).

More specifically, the following models are trained: unconditional flow matching (baseline), conditional flow matching with label conditioning (direct conditioning), classifier guidance (separate classifier), and classifier-free guidance with label dropout. This produces 12 trained models in total. Where applicable, each model type is evaluated using the same guidance scales and sampling procedures as those used in the earlier research questions, ensuring comparable measurements. For each dataset and model type, FID, precision, and recall are computed.

By comparing these metrics across datasets, this experiment reveals how dataset complexity affects the relative performance of different guidance mechanisms. Patterns in optimal guidance scales indicate whether simpler or more complex datasets benefit more from guidance. Comparing failure modes identifies which datasets and which guidance methods are most susceptible to quality degradation.

3.3.5. Additional Investigation: Understanding Guidance Failure

The results will show that classifier guidance can lead to worse samples than unconditional generation for certain datasets and at certain guidance scales. To further explore this observation, two additional investigations were conducted.

First, the classifier performance is analysed by evaluating classification accuracy at different image-noise interpolations. For each dataset, noisy interpolation images are created at time points $t \in [0, 1]$ and classification accuracy is measured. This reveals where the classifier becomes unreliable, explaining why guidance fails when the classifier cannot distinguish classes in noisy samples.

Second, three sampling strategies are compared to isolate the source of performance differences: (1) unconditional generation, which samples $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and simulates the ODE without class control, (2) classifier guidance at $w = 0$, which uses the same unconditional vector field but generates equal numbers per class, and (3) classifier-free guidance at $w = 0$, which produces conditional samples without guidance amplification. By comparing FID, precision, and recall across these three approaches, this experiment separates two factors: the effect of balanced class sampling versus the quality of the learned vector field. If classifier guidance at $w = 0$ performs differently from unconditional generation, this indicates that balanced sampling alone affects metrics. If it differs from CFG at $w = 0$, this reveals differences in how the two methods learn vector fields. This controlled comparison may explain the mechanisms behind guidance failure.

Chapter 4

Results

Having discussed the experimental methodology, the findings are now organized around four research questions. Each section presents results addressing a single research question, followed by an analysis and interpretation of the observed patterns.

4.1. Research Question 1: Effect of Label Conditioning

FID, precision, and recall scores for models trained in unconditional and basic label conditioning are shown in Table 4.1.

Table 4.1: Unconditional vs label-conditional generation.

Method	MNIST			SVHN			CIFAR-10		
	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑
Unconditional	22.22	0.50	0.72	41.65	0.59	0.64	52.96	0.82	0.21
Label Conditioning	2.45	0.75	0.78	11.26	0.75	0.67	14.87	0.80	0.55

Consistent and substantial improvements are observed across all datasets when adding label conditioning. On MNIST, FID reduces from 22.22 to 2.45, representing a 9-fold improvement. SVHN shows FID reducing from 41.65 to 11.26, a 3.7-fold improvement. CIFAR-10 improves from 52.96 to 14.87, a 3.6-fold improvement. The magnitude of improvement correlates inversely with dataset complexity. CIFAR-10 shows a smaller relative improvement, possibly because even with class labels, the model must capture substantial within-class variation, such as different dog breeds or car angles.

An interesting pattern emerges in CIFAR-10. The unconditional model achieves high precision (0.82) but very low recall (0.21), indicating mode collapse. The model generates realistic-looking samples but only covers a small subset of the data distribution. Label conditioning addresses this problem, improving recall from 0.21 to 0.55 while maintaining precision at 0.80, which aligns with expectations. Label conditioning thus enables the model to better capture the diversity within each class and to avoid collapsing into a few dominant modes across all classes.

4.2. Research Question 2: Impact of Classifier Guidance

The experiments described in Section 3.3.2 investigate whether adding classifier gradients to the vector field improves generation quality.

FID, precision, and recall scores for models trained with classifier guidance across guidance scales are shown in Table 4.2.

Table 4.2: Classifier guidance results across guidance scales.

Guidance Scale w	**MNIST**			**SVHN**			**CIFAR-10**		
	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑
0.0	2.59	0.73	0.76	11.46	0.75	0.67	17.79	0.78	0.53
1.0	9.58	0.54	0.81	64.07	0.38	0.64	43.91	0.39	0.11
2.0	17.69	0.44	0.83	73.55	0.27	0.57	53.45	0.28	0.10
4.0	15.31	0.47	0.82	83.47	0.16	0.12	64.51	0.20	0.04
6.0	18.92	0.41	0.80	92.94	0.11	0.01	69.44	0.19	0.02

Classifier guidance consistently results in lower-quality samples. Best results always occur at $w = 0$, where no guidance is applied. On MNIST, FID degrades from 2.59 to 17.69 at $w = 2$, and classifier guidance is even worse on SVHN.

Figure 4.1 illustrates this quality-diversity trade-off across all datasets. The results differ from the literature on diffusion models [10], where classifier guidance typically improves precision while reducing recall. Here, the opposite is observed for MNIST: precision decreases while recall increases. On MNIST, precision drops from 0.73 to 0.44 while recall increases from 0.76 to 0.83. For SVHN and CIFAR-10, both precision and recall decrease, indicating guidance failure, with samples becoming both less realistic and less diverse. This reversed trade-off for MNIST suggests that classifier gradients push generation toward atypical regions rather than toward class prototypes, as further investigated in Section 4.5.

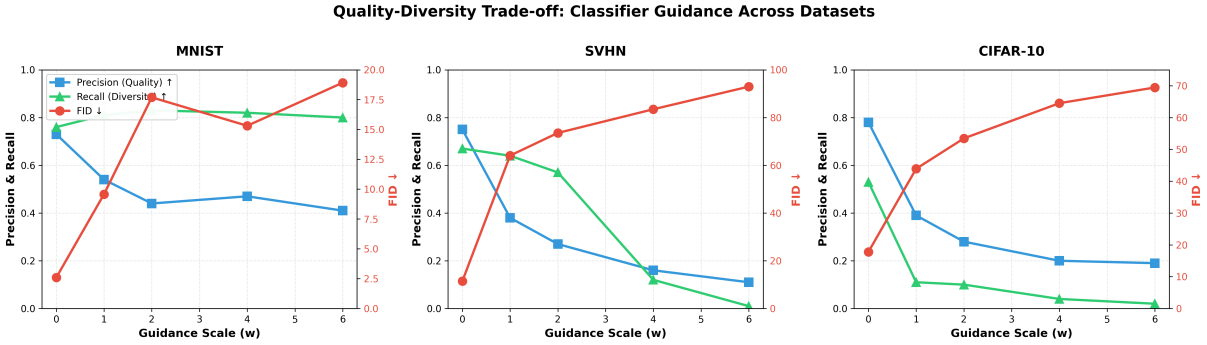

Figure 4.1: Quality-diversity trade-off with classifier guidance across all three datasets. As the guidance scale increases, precision decreases while recall increases for MNIST (reversed trade-off), but both precision and recall decrease for SVHN and CIFAR-10 (indicating guidance failure).

4.3. Research Question 3: Impact of Classifier-Free Guidance

Classifier-free guidance is implemented as an alternative that requires only a single model, as described in Section 3.3.3.

The influence of different guidance scales in classifier-free guidance is shown by reporting the FID, precision, and recall in Table 4.3.

Table 4.3: Classifier-free guidance results across guidance scales.

Guidance Scale w	**MNIST**			**SVHN**			**CIFAR-10**		
	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑
0.0	3.53	0.67	0.80	11.33	0.75	0.68	27.54	0.77	0.45
1.0	2.43	0.77	0.77	11.29	0.75	0.68	19.84	0.80	0.47
2.0	3.83	0.84	0.65	21.14	0.77	0.65	14.18	0.85	0.42
4.0	10.08	0.85	0.50	50.16	0.76	0.63	27.54	0.82	0.23
6.0	16.91	0.82	0.40	74.80	0.71	0.62	40.66	0.74	0.11

Classifier-free guidance illustrates the quality-diversity trade-off observed in the diffusion model literature. On MNIST, optimal FID occurs at w = 1.0 with a value of 2.43. As guidance increases from w = 0 to w = 4, precision improves from 0.67 to 0.85 while recall decreases from 0.80 to 0.50.

CIFAR-10 shows the strongest benefit from CFG. Optimal FID occurs at $w = 2.0$ with FID 14.18, performing similarly to label conditioning (14.87 from Table 4.1) and outperforming classifier guidance at any scale (Table 4.2). Precision increases from 0.77 at $w = 0$ to 0.85 at $w = 2$ while recall decreases modestly from 0.45 to 0.42.

Figure 4.2 illustrates the quality-diversity trade-off across all datasets.

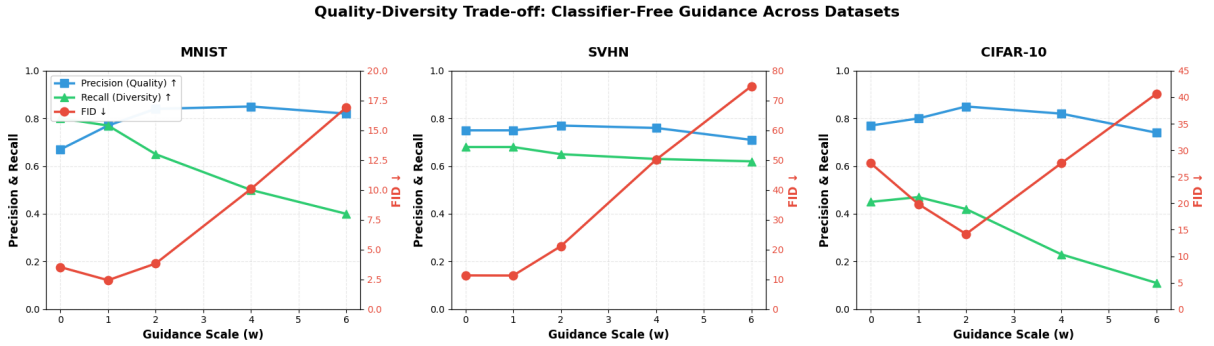

Figure 4.2: Quality-diversity trade-off with classifier-free guidance. As the guidance scale increases, precision improves while recall decreases for all three datasets. The left y-axis shows precision and recall, while the right y-axis shows FID.

4.4. Research Question 4: Cross-Dataset Comparison

FID, precision, and recall scores for the best-performing models in each of the previous experiments are presented in Table 4.4 in order to compare the sampling ability across all datasets.

Table 4.4: Best performing configuration for each method across datasets.

Dataset	Method	Best w	FID ↓	Precision ↑	Recall ↑
MNIST	Unconditional	–	22.22	0.50	0.72
	Label Conditioning	–	2.45	0.75	0.78
	Classifier Guidance	0.0	2.59	0.73	0.76
	CFG	1.0	2.43	0.77	0.77
SVHN	Unconditional	–	41.65	0.59	0.64
	Label Conditioning	–	11.26	0.75	0.67
	Classifier Guidance	0.0	11.46	0.75	0.67
	CFG	1.0	11.29	0.75	0.68
CIFAR-10	Unconditional	–	52.96	0.82	0.21
	Label Conditioning	–	14.87	0.80	0.55
	Classifier Guidance	0.0	17.79	0.78	0.53
	CFG	2.0	14.18	0.85	0.42

Figure 4.3 provides visual evidence supporting these quantitative findings. The figure shows generated samples of class 0 (digit "0" for MNIST and SVHN, 'airplane' for CIFAR-10) across all conditioning methods, illustrating how dataset complexity affects generation quality.

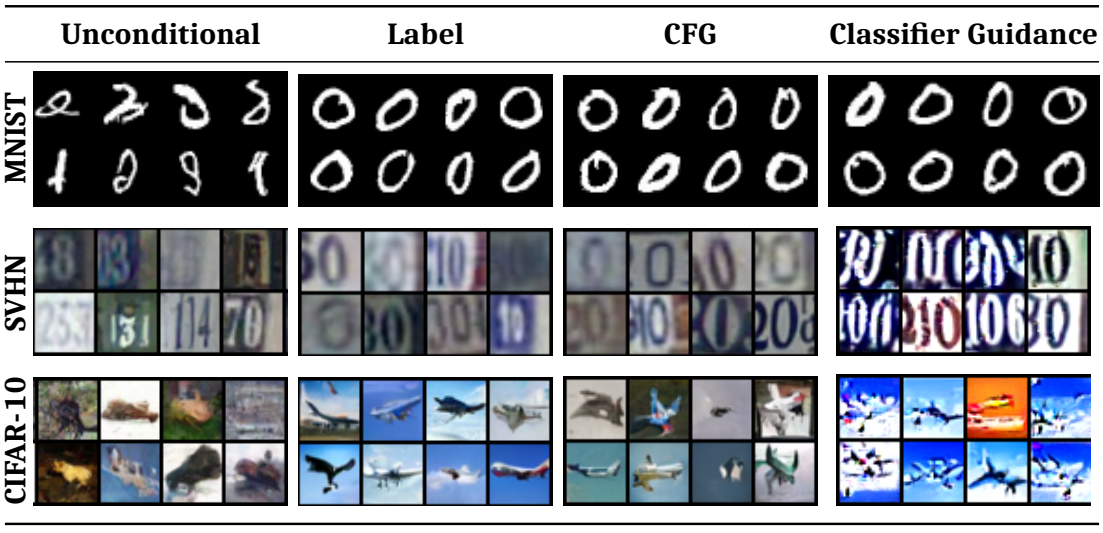

Figure 4.3: Generated samples of class 0 (digit "0" for MNIST and SVHN, 'airplane' for CIFAR-10) from CFM models using different conditioning methods. Each cell displays a 2×4 grid of randomly generated samples. The visual quality degradation from MNIST to CIFAR-10 reflects the increasing dataset complexity and the corresponding challenge in modeling within-class variation.

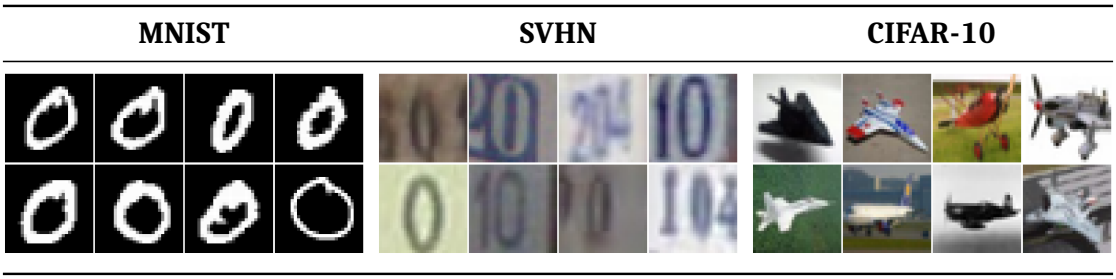

Figure 4.4: Real training examples of class 0 from each dataset (digit "0" for MNIST and SVHN, 'airplane' for CIFAR-10). Each cell displays a 2×4 grid of randomly selected training samples to serve as a visual reference for comparing with the generated samples in Figure 4.3.

The visual samples in Figure 4.3 support the quantitative results shown in Table 4.4. Comparing with real examples in Figure 4.4, for MNIST, all conditioning methods produce recognizable digit "0" samples with clear circular structures, and are consistent with the high precision scores (0.75–0.77). The similarity in visual quality across label conditioning, classifier-free guidance, and classifier guidance is reflected in their comparable FID scores (2.43–2.59), confirming that conditioning nearly solves the generation problem for simple grayscale digits.

For SVHN, the quality gap between unconditional and conditioned methods becomes more pronounced. The unconditional model produces blurry, low-contrast samples (precision: 0.59, recall: 0.64), while conditioned methods generate sharper, more recognizable house-number "0" digits (precision: 0.75, recall: 0.67–0.68). However, even the best methods struggle more than with MNIST, as evidenced by higher FID scores (11.26–11.46) and visual artifacts in some samples, reflecting the increased dataset complexity.

The impact of dataset complexity is most evident in CIFAR-10 results. The unconditional model produces vague, barely recognisable samples, demonstrating severe mode collapse with low recall (0.21) despite deceptively high precision (0.82). Label conditioning and classifier-free guidance show improvement, generating coherent airplanes with visible wings and fuselages, corresponding to dramatically improved recall scores (0.42–0.55). However, even the best-performing CFG method (FID: 14.18, precision: 0.85, recall: 0.42) shows some limitations in capturing within-class variation, as some generated airplanes lack fine structural details. This visual evidence suggests that difficulty remains in modeling complex natural images, even with sophisticated conditioning approaches.

Notably, classifier guidance performance degrades visually across datasets, particularly evident in the SVHN and CIFAR-10 samples, where some generated images show artifacts or inconsistencies. This visual degradation correlates with a decrease in the classifier's accuracy across datasets (MNIST: 95%, SVHN: 85%, CIFAR-10: 50.43%), possibly indicating that classifier guidance fails when the classifier cannot correctly classify noisy interpolated images.

These results align with expectations regarding dataset complexity. Furthermore, it has been observed that more complex datasets benefit from stronger guidance. Classifier guidance failure worsened with dataset complexity, correlating with classifier accuracy. CFG performed consistently well across all datasets, matching or beating all other methods at their optimal scales. Unlike simple label conditioning, CFG allows dynamic control over conditioning strength during generation. CFG also has the benefit of not requiring an additional classifier.

4.5. Discussion on Guidance Failure

Guidance at $w = 0$ produced better samples than an unconditional model (the baseline), and sample quality degraded as the guidance scale increased for classifier guidance.

Classifier guidance at $w = 0$

At $w = 0$, Equation 2.21 gives:

$$\boldsymbol{u}_{\text{guided}}(\boldsymbol{x}_t, t, y) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t) + 0 \cdot \nabla_{\boldsymbol{x}_t} \log p_\phi(y|\boldsymbol{x}_t, t) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t), \tag{4.1}$$

and is equivalent to the unconditional generation case. Yet classifier guidance at $w = 0$ achieves FID 2.59 on MNIST (Table 4.2) while an unconditional model (the baseline) gets 22.22 (Table 4.1).

A salient difference is not in how generation happens, but in how classes are sampled. Unconditional generation produces samples from whichever class distribution the model learned, and may be imbalanced. For classifier guidance at $w = 0$, roughly balanced sampling is done by generating equal numbers per class and using the classifier to verify that they match the target class. This roughly balanced sampling might improve FID because it better matches the test set's roughly balanced distribution. Further experimentation is needed to better understand this result.

Classifier guidance when $w > 0$

When classifier gradients are added with $w > 0$, generation quality degrades because the classifier performs poorly on noisy interpolated images. The degradation correlates with classifier accuracy across datasets. MNIST with 95% accuracy showed moderate degradation, SVHN with 85% accuracy showed severe degradation, and CIFAR-10 with 50.43% accuracy showed the worst performance. Classifier performance across flow-interpolated timesteps from pure noise ($t = 0.0$) to clean images ($t = 1.0$) for all three datasets is shown in Figure 4.5 and Table 4.5.

Table 4.5: Classifier performance across flow-interpolated timesteps from pure noise ($t = 0.0$) to clean images ($t = 1.0$) for all three datasets.

Dataset	0.0 (noise)	0.2	0.4	0.6	0.8	1.0 (clean)
MNIST (95%)	0.1068	0.4951	0.9318	0.9869	0.9898	0.9889
SVHN (85%)	0.1959	0.2570	0.5006	0.7379	0.8570	0.8879
CIFAR-10 (50.43%)	0.1023	0.3129	0.4955	0.6121	0.6805	0.6934

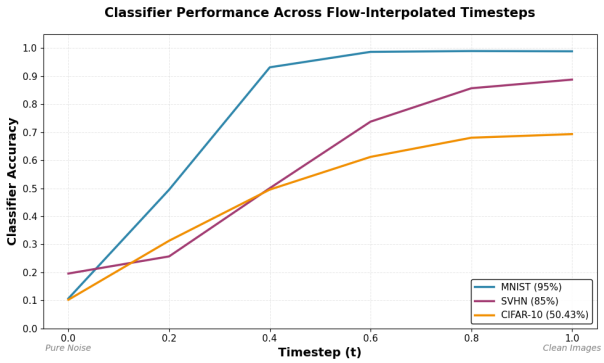

Figure 4.5: Classifier accuracy across flow-interpolated timesteps from noise ($t = 0$) to clean images ($t = 1$).

During generation, the ODE solver traverses the entire trajectory from $t = 0$ (noise) to $t = 1$ (data). At each timestep, classifier guidance computes gradients $\nabla_{\mathbf{x}_t} \log p_\phi(y|\mathbf{x}_t, t)$ and adds them to the vector field. When the classifier cannot accurately classify $\mathbf{x}_t$ at timestep t, as evidenced by low accuracy at that timestep, these gradients may point toward potentially misleading features rather than genuine class characteristics. The model follows these gradients, pushing samples away from the data manifold rather than toward class prototypes.

Classifier free guidance at $w = 0$

It is worth emphasizing that CFG at $w = 0$ still does direct label conditioning. At $w = 0$, Equation 2.25 gives:

$$\boldsymbol{u}_{\text{cfg}}(\boldsymbol{x}_t, t, y) = (1 + 0) \cdot \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, y) - 0 \cdot \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, \text{null}) = \boldsymbol{u}_\theta(\boldsymbol{x}_t, t, y), \tag{4.2}$$

as in the case of direct label conditioning.

During CFG training, labels are dropped 10% of the time and replaced with a null token—an extra embedding (the $(K + 1)$-th embedding for K classes). When the model sees the null token, it learns to generate unconditionally. When it sees a real class token, it learns to generate that class.

At test time with $w = 0$, real class labels are used (not the null token), meaning conditional generation is performed. This explains why CFG at $w = 0$ gets FID 3.53 (Table 4.3), close to label conditioning at 2.45 (Table 4.1), instead of 22.22 as in unconditional.

Chapter 5

Conclusion

Guidance has been an effective conditioning mechanism in diffusion models, leading to improved samples by balancing sample fidelity and diversity. Given the differences between diffusion (described by stochastic paths) and flow models (described by deterministic paths), this report investigated whether guidance mechanisms could improve conditional generation in flow models. Flow models were trained with four approaches — unconditional generation, label conditioning, classifier guidance, and classifier-free guidance — on three datasets of increasing complexity: MNIST, SVHN, and CIFAR-10.

Label conditioning improved results across all datasets, as measured by FID, with the largest gains observed on simpler datasets. Conditioning allows the model to learn class-specific structures, but the benefit diminishes as dataset complexity increases. For simpler datasets, label conditioning nearly resolves the generation problem. For the more complex datasets, intra-class variation remains challenging for the models explored.

Classifier guidance did not yield any improvement in sample quality compared to label conditioning, presumably because training the classifier on noisy interpolations is difficult. Classifier guidance, however, leads to better results than unconditional generation. This may be due to the approach of uniformly sampling over all classes at test time. Unlike diffusion models, where classifier guidance improves quality at the expense of diversity, flow matching exhibited the opposite pattern: sample quality declined while diversity remained stable or even increased. Again, this observation may be related to the challenges encountered with the classifier, suggesting a limitation that could be improved in the future.

An interesting difference to the diffusion model literature is that optimal guidance scales were much lower for flow models. Understanding why would require a more thorough study, including reproducing diffusion models under the same experimental procedure.

Classifier-free guidance showed the expected quality-diversity tradeoff across all datasets. As guidance strength increased, sample quality improved while diversity decreased. Its success may be due to joint training on the generative process itself, rather than reliance on an external classifier, which can be difficult to train.

5.1. Limitations and Future Work

This study has several limitations that suggest directions for future research.

FID, precision, and recall were computed from 10,000 generated samples. Increasing the sample size to 50,000 would provide more stable results, and is in line with the literature [10]. Extending the evaluation to high-resolution datasets such as CelebA-HQ (1024×1024) or ImageNet (256×256) would test whether the observed behaviors generalize to more realistic image generation tasks. Testing alternative architectures, such as Vision Transformers, could reveal whether guidance performance depends on architectural design. Similarly, increasing the number of ODE integration steps beyond 50 may improve generation quality, especially for complex datasets.

Future work could explore continuous or multimodal conditioning, such as text-based prompts via CLIP [28] embeddings, to assess the flexibility of guidance mechanisms.

The linear interpolation scheme was not varied. It would be interesting to explore alternative transport schemes, such as affine coupling layers or learned optimal transport maps, which might exhibit different behavior with guidance mechanisms.

The classifiers employed standard architectures that performed poorly on noisy interpolations. Developing noise-aware classifiers with conditional normalization, sinusoidal time embeddings, or progressive denoising training may improve robustness.

Bibliography

[1] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *International Conference on Learning Representations*, 2023.

[2] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" *arXiv preprint*, 2019.

[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

[4] Y. Yang, S. Mandt, and L. Theis, "An introduction to neural data compression," *Foundations and Trends in Computer Graphics and Vision*, vol. 15, no. 2, pp. 113–200, 2023.

[5] L. Theis, T. Salimans, M. D. Hoffman, and F. Mentzer, "Lossy compression with Gaussian diffusion," *arXiv preprint*, 2022.

[6] J. Yoon, J. Jordon, and M. v. d. Schaar, "GAIN: Missing data imputation using generative adversarial nets," in *International Conference on Machine Learning*, 2018.

[7] R. D. Camino, C. A. Hammerschmidt, and R. State, "Improving missing data imputation with deep generative models," *arXiv preprint*, 2019.

[8] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," in *ACM SIGGRAPH*, 2022.

[9] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 2020.

[10] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Advances in Neural Information Processing Systems*, 2021.

[11] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.

[12] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Advances in Neural Information Processing Systems*, 1995.

[13] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2019.

[14] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[15] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "FFJORD: Free-form continuous dynamics for scalable reversible generative models," in *International Conference on Learning Representations*, 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[17] Y. Wu and K. He, "Group normalization," in *European Conference on Computer Vision*, 2018.

[18] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *Advances in Neural Information Processing Systems*, 2022.

[19] P. Holderrieth and E. Erives, "Introduction to flow matching and diffusion models," 2025, online tutorial, MIT CSAIL.

[20] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *AAAI Conference on Artificial Intelligence*, 2018.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[25] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017.

[26] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila, "Improved precision and recall metric for assessing generative models," in *Advances in Neural Information Processing Systems*, 2019.

[27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[28] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, 2021.