An Empirical Study of Flow Matching and Comparison with DDPM for Image Generation

Oscar Hoffmann
oscho091@student.liu.se

Fredrik Ramberg
frera064@student.liu.se

William Rimton
wilri858@student.liu.se

Erik Bertilsson
eribe255@student.liu.se

Henrik Björkander
henbj286@student.liu.se

Abstract

This report investigates the implementation and performance of a conditional Flow Matching model (FM), based on the 2022 paper *Flow Matching for Generative Modeling* (Lipman et al., 2022). The model can be used for generating images and is an alternative to the widely used Denoising Diffusion Probabilistic Models (DDPM).

A Conditional Flow Matching model with Optimal Transport (OT) was trained on the CIFAR-10 dataset, which contains 32x32 resolution RGB images (Krizhevsky, 2009). The model was evaluated using the Frechet Inception Distances (FID) score (Seitzer, 2020). Despite the limited computer resources, the result indicated a well-performing model according to the evaluation metrics.

In comparison to the DDPM model, which achieved a FID score of approximately 55, the Flow Matching model attained a FID score of around 49 within the same training duration. The report demonstrates that the FM model performs better in quality and speed in inference with the same training time as the DDPM. The findings suggest that Flow Matching with OT is a promising technique that warrants further exploration and application to enhance the performance of both small and large-scale image generators.

1 Introduction

Generative deep learning image models, which aim to transform data between different distributions, have recently revolutionized generative modeling. Recent advances have already seen broad application in industry and among consumers. As the performance of deep learning models evolve they are becoming increasingly more expensive and time-intensive to train and use, hence methods to reduce training times while increasing inference performance are sought-after. One such area is in the field of generative images.

The report explores Flow Matching, a new technique that aims to reduce time intensity for training models built on Continuous Normalizing Flows (CNF) for image generation. In contrast to DDPM, CNF replaces iterative noise predictions with a continuous vector field generated by continuous differential equations which can improve inference quality and speed. This report aims to implement a Flow Matching model with OT and compare the performance with DDPM.

2 Related work

Continuous Normalizing Flows (CNF) were first introduced in Chen et al., 2018. It proposed expanding Normalizing Flows, which use discrete layers of transformations, with a continuous flow which results in a more adaptable model. Although more flexible, they required solving ordinary differential equations for training which added time complexity and made them inefficient to train.

Flow matching was introduced in Lipman et al., 2022 as a simulation-free approach for training Continuous Normalising flows (CNF) based on regressing vector fields of fixed conditional probability paths. Instead of specifying the loss function as the comparison of the generated images, the model will compute the loss based on the difference in the predicted vector field of the flow, hence the name Flow Matching. This enables bypassing the time-intensive generation step with the ODE-solver in training drastically reducing training times. Furthermore, the authors use Optimal Transport as a way to define the straight path of the flow in order to further simplify the training process. The article produced promising results for the Flow Matching model which could be trained with ease.

Further explorations of Conditional Flow Matching with OT are made in Tong et al., 2023 which produces good results for the technique. The technique of Flow Matching shows great promise and has already been applied in some commercial image generators such as Stable Diffusion 3 (Esser et al., 2024).

3 Problem formulation

This report implemented Flow Matching with Optimal Transport and evaluated its performance and speed. This was done by training a conditional Flow Matching model on the CIFAR-10 dataset (Krizhevsky, 2009) and evaluating the inference with Fréchet Inception Distance (FID) score (Seitzer, 2020). Furthermore, a conditional DDPM was trained using the same data set as a benchmark in the Flow Matching evaluation. The aim was to contrast the performance of an FM model with a more traditional model to explore potential advantages and disadvantages.

4 Method

The following section will describe the theory behind the Flow Matching model used in our experiments. It is based on the model suggested in Lipman et al., 2022. Necessary proofs for this section can be found in appendix 9.

4.1 Flow matching

The main objective of Flow Matching (FM) is to train a neural network to learn a continuous vector field $v_t(x)$ as an approximation of a target vector field $u_t(x)$. This target vector field $u_t(x)$ dictates how each data point x at a given time t should traverse from the initial state to the target distribution.

In the context of Flow Matching, x_0 is defined as a random variable sampled from the initial input space, and x_1 is a random variable from, here a data sample, from the unknown target distribution state $q(x_1)$.

Flow Matching optimizes Continuous Normalizing Flow (CNF) models. The basics of CNF are that for time $t \in [0, 1]$ we have the probability path $p_t(x)$ such that $p_0 = p$ is a simple Gaussian-distribution and $p_1 = q$. Thereby constructing a path $p_t(x)$ that models a flow from p_0 to p_1. The flow is then defined as $\phi_t(x)$ for a given data point x and time t, where:

$$\frac{\mathrm{d}\phi_t(x)}{\mathrm{d}t} = v_t(\phi_t(x)) \tag{1}$$

$$\phi_0(x) = x \tag{2}$$

From this, the probability density at time t can be determined by the push-forward equation as follows:

$$p_t = [\phi_t]_* p_0 \tag{3}$$

4.2 Conditional probability paths

To construct a target probability path we use a general family of Gaussian conditional probability path $p_t(x|x_1)$ shown below. However, we do not know the distribution of these conditional probability paths beforehand.

$$p_1(x|x_1) = \mathcal{N}(x|\mu_t(x_1), \sigma_t(x_1)^2\mathcal{I}) \tag{4}$$

Where μ_t is the time-dependent mean and σ_t the time-dependent standard deviation. We then consider the conditional flow ψ and conditional vector field $u_t(\psi_t(x)|x_1)$ such that:

$$\frac{\mathrm{d}\psi_t(x)}{\mathrm{d}t} = u_t(\psi_t(x)|x_1) \tag{5}$$

This leads to the conditional vector field $u_t(\psi_t(x)|x_1)$ which generates the set of gaussian paths $p_t(x|x_1)$.

$$u_t(\psi_t(x)|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}(\psi_t(x) - \mu_t(x_1)) + \mu_t'(x_1) \tag{6}$$

4.3 Optimal Transport

There are many flows possible from the source to the target distribution. For this report we use a standard Gaussian noise distribution where $\mu_0(x_1) = 0$ and $\sigma_0(x_1) = 1$ whilst $\mu_1(x_1) = x_1$ and $\sigma_1(x_1) = \sigma_{\text{min}}$. Optimal transport simply defines that the probability path should consist of a straight trajectory from $p_0(x_0)$ to $q(x_1)$. We change the above equation 6, and express it accordingly.

$$u_t(\psi_t(x)|x_1) = \frac{x_1 - (1 - \sigma_{min})}{1 - (1 - \sigma_{min})t}\psi_t(x) \tag{7}$$

The conditional flow for optimal transport will then be:

$$\psi_t(x) = (1 - (1 - \sigma_{\text{min}})t)x + tx_1 \tag{8}$$

4.4 Loss function

The loss function is presented below where $v_t(\psi_t(x_0))$ is the vector field v_t of the flow ψ of x_0 at time t. Essentially, the loss regresses the conditional vector field $u_t(\psi_t(x)|x_1)$ with a neural network v_t of the conditional flow $\psi_t(x)$, which will learn the model to generate $p_t(x)$.

$$L_{CFM}(\theta) = E_{t,q(x_1),p(x_0)}\|v_t(\psi_t(x_0)) - (x_1 - (1 - \sigma_{\text{min}})x_0)\|^2 \tag{9}$$

4.5 Training algorithm

The table below describes the training algorithm used for DDPM and Flow Matching, highlighting how they differentiate.

```
Flow Matching
Input: dataset $q$, noise $p$, batch $b$ label $l$
Initialize neural network model $v^\theta$
For $b$ in $q$ do
    $t \sim U([0,1]) \rightarrow$ sample time
    $x_1, l_1 \sim b(x_1) \rightarrow$ sample data with corresponding label
    $x_0 \sim p(x_0) \rightarrow$ sample noise
    $x_t = \psi_t(x_0|x_1) \rightarrow$ conditional flow
    Gradient step with $\nabla\|v_t^\theta(x_t, l_1) - \dot{x}_t\|^2$
Output: $v^\theta$
```

```
Diffusion
Input: dataset $q$, noise $p$, batch $b$, label $l$
Initialize neural network model $m$
For $b$ in $q$ do
    $t \sim U([0,1000]) \rightarrow$ sample time
    $x_1 \sim q(x_1) \rightarrow$ sample data
    $p_t \sim p(x_t|x_1) \rightarrow$ sample noise
    $x_t = f(x_1, p_t) \rightarrow$ add noise to data
    $pred = m(t, x_t, l) \rightarrow$ predict noise
    Gradient step with $\nabla\|pred - p_t\|^2$
Output: $m$
```

Table 1: Training: Flow Matching and Diffusion

5 Experiments

This section outlines the computational framework and settings employed in our experiments. The study utilized a U-Net-based architecture for both the Flow Matching and DDPM model. The DDPM model used a beta scheduler from 0.0001 to 0.02, 1000 steps, and mean squared error as the loss function. In order to compare the inference time between the models, 100 images were generated with both models. The hardware used for training and inference in this study was a consumer-grade computer with a 4060Ti 16GB VRAM GPU, 32GB RAM, and Intel i7 8700K CPU.

5.1 Dataset

The models were trained on the CIFAR-10 dataset (Krizhevsky, 2009). This dataset contains 60 000 32x32 pixel images together with labels for ten different categories. The categories are *airplane, automobile, bird, cat, deer, dog, frog, horse, ship* and *truck*. The data is divided into 50 000 training images and 10 000 test images.

5.2 Optimizer and Parameters

The Adam optimizer was employed for both models with the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay of 0, and $\epsilon = 1e{-}8$.

Parameter	**Flow Matching**	**DDPM**
Batch Size	128	128
Epochs	500	500
Learning Rate	1e-3	1e-3
Learning Rate Scheduler	Polynomial Decay	Polynomial Decay

Table 2: Hyperparameters used in training the models.

5.3 Evaluation Metrics

To quantitatively assess the quality of the generated images, FID was used as the evaluation metric. FID measures the similarity and distance between the distribution of generated images and the distribution of real images such that a lower score indicates that the two distributions are more similar. Thus, a lower score is indicative of better model performance.

$$FID = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}(\sigma_r + \sigma_g - 2\sqrt{\sigma_r\sigma_g}) \tag{10}$$

Where μ_r and μ_g are the mean vectors of the feature distributions extracted from the real and generated images, respectively, and σ_r and σ_g are the covariance matrices from respective feature distribution. A further detailed explanation of FID is provided in (Section 9.5).

The training and performance characteristics of the evaluated models are presented below. Note that some metrics, such as training and inference speed, are specific to the hardware detailed in Section 5.

5.4 Training and Inference Time

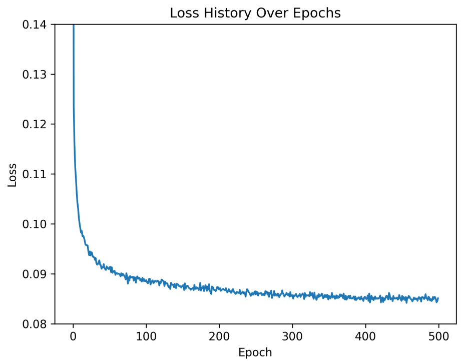
The training time for the DDPM model was 18 hours, 21 minutes, and 16 seconds. In comparison, the Flow Matching model had a slightly longer training time of 18 hours, 31 minutes, and 14 seconds. To evaluate the speed of inference 100 images were generated to calculate the average time to generate one image for both models, which was 1013ms for DDPM and 804ms for Flow Matching. Samples of the generated images from both models can be found in Appendix A (Section 9).

Model	Training time	Inference time
Flow Matching	18h, 31min, 14s	804ms
DDPM	18h, 21min, 16s	1013ms

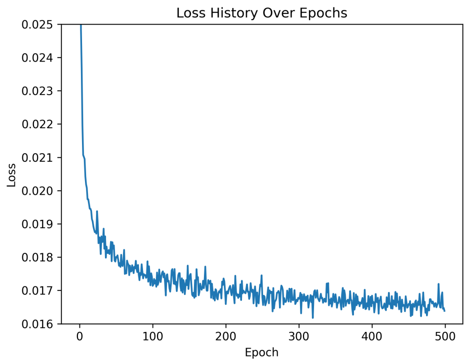
Table 3: Training time for 500 epochs and inference time for one image

5.5 Loss History

The loss history for Flow Matching is plotted in Figure 1a and the loss history for DDPM is plotted in Figure 1b. Note that the models used different loss functions, meaning that the absolute loss values cannot be directly compared.



(a) Loss history for Flow Matching



(b) Loss history for DDPM

Figure 1: Loss history

5.6 FID Score

The FID scores were calculated every 50 epochs and are plotted in Figure 2. Due to hardware limitations, the evaluation was conducted using 1500 generated images, compared to 1500 real images from the CIFAR-10 test dataset, as mentioned in Section 5.3.

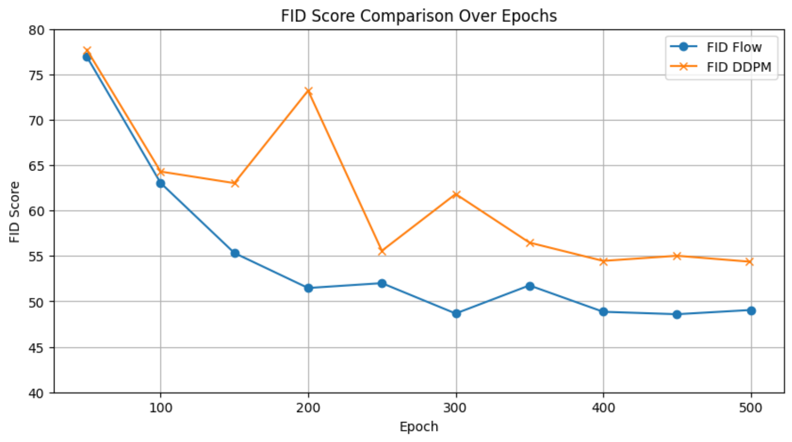


Figure 2: FID Score comparison over epochs

6 Discussion & Conclusion

Flow Matching, in contrast to diffusion models, directly transforms data from a source to a target distribution using an optimal flow path. This approach results in faster and more efficient inference, as demonstrated in Table 3. However, training Flow Matching was slightly slower compared to DDPM in our implementation, as shown in Table 3. It is worth noting that if more steps were used during training for DDPM, to achieve matching FID scores, it is possible that Flow Matching would be faster in training as well.

The evaluation showed that Flow Matching achieved better FID scores than DDPM consistently over the epochs, indicating superior image generation quality given the respective implementations (Figure 2). This suggests that Flow Matching captures the data distribution effectively. While the generated images in Appendix 9 do not show any significant difference in image quality between the two models, they indicate that the FID scores are reasonable.

The loss history for both models show stable training, with Flow Matching appearing slightly more stable (Figure 1). Both models converged well, indicating effective learning.

In conclusion, Flow Matching demonstrates significant potential as a robust method for image generation. Flow Matching shows promise as an alternative to DDPM, offering faster inference and better FID scores given the respective implementations. Future studies could explore other implementations of Flow Matching to further improve performance.

7 Ethical Discussion

There are several ethical implications when comparing Flow Matching and DDPM. As mentioned in the chapter *Related work* 2, Flow Matching offers potentially faster training and sampling speeds compared to DDPM. This presents an ethical advantage in terms of reduced energy consumption, aligning it with sustainability goals in AI.

Generative image models have the potential to produce highly realistic images, which raises ethical concerns regarding the generation of "deep fakes". This capability highlights the need for guidelines to prevent misuse in generating fraudulent content. Furthermore, transparency in model training is essential to ensure that biases are minimized. Finally, the accessibility of image generating models can democratize media creation and increase creative freedom.

8 Contribution statement

1. Henrik: Implemented FID Score in code and general code structuring. Written section 5.2, 5.3 in this report, appendix C on FID Score. Written about theory in section 4 and deriving proofs in Appendix B.

2. Erik: Implementing Flow matching in code, training loop for flow matching, solving the ode in code, and connecting to Unet in code. Theory in section 4 and deriving equations in Appendix B.

3. Oscar: Implemented DDPM and trained both models. Implemented polynomial decay learning rate scheduler, timers, loss history- and FID score trackers for both models. Plotted graphs and debugged FID score code. Wrote sections 5.4, 5.5, 5.6 + Discussion & Conclusion, Ethical Discussion, and Appendix A.

4. Fredrik: Implemented dataset handling as well as aided in checkpoint handling and ddpm debugging. Wrote Abstract, Introduction, Related Works, Problem formulation as well as contributions to the Flow Matching section.

5. William: Implemented DDPM and debugging in code. Written Method, Experiments as well as contributions to Appendix B in this report.

References

Chen, Ricky TQ, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud (2018). “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31.

Esser, Patrick, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. (2024). “Scaling rectified flow transformers for high-resolution image synthesis”. In: *arXiv preprint arXiv:2403.03206*.

Krizhevsky, Alex (2009). *CIFAR-10 (Canadian Institute for Advanced Research)*. Online. Accessed: 2024-05-12. URL: `https://www.cs.toronto.edu/~kriz/cifar.html`.

Lipman, Yaron, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le (2022). “Flow matching for generative modeling”. In: *arXiv preprint arXiv:2210.02747*.

Seitzer, Maximilian (2020). *pytorch-fid: FID Score for PyTorch*. `https://github.com/mseitzer/pytorch-fid`. Version 0.3.0.

Tong, Alexander, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio (2023). “Improving and generalizing flow-based generative models with minibatch optimal transport”. In: *arXiv preprint arXiv:2302.00482*.

9 Appendix

Appendix A: Real and Generated Images

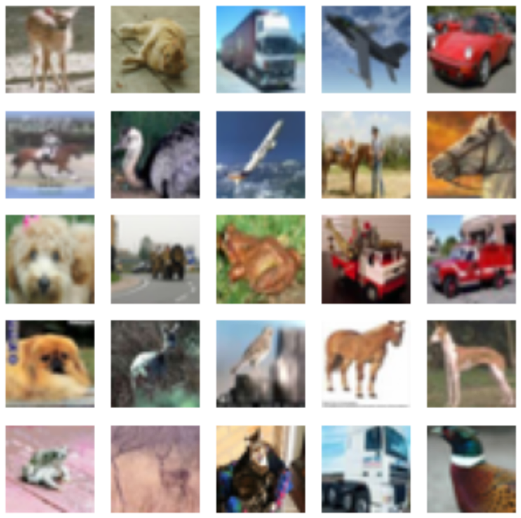
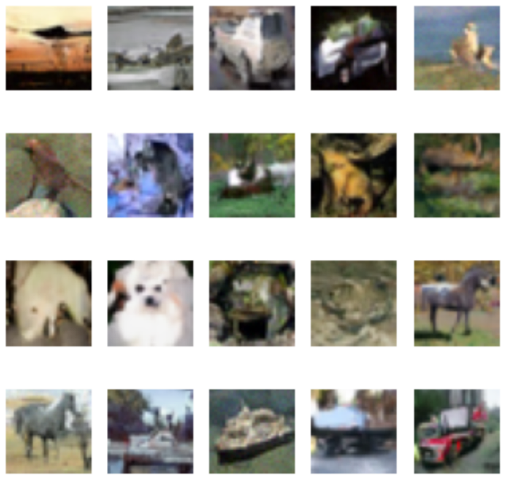
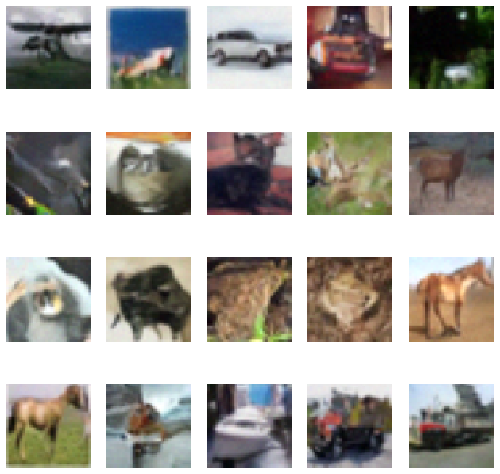


Figure 3: Sample images from the CIFAR-10 training data set (real)



(a) Generated images from Flow Matching



(b) Generated images from DDPM

Figure 4: Sample of images from the respective models with two images per class (generated)

Appendix B: Derivation of formulas

9.1 Push-forward Equation

The push forward equation is defined as

$$[\phi_t]_* p_0(x) = p_0(\phi_t^{-1}(x)) \cdot \det\left(\frac{\partial \phi_t^{-1}}{\partial x}(x)\right) \tag{11}$$

9.2 Conditional Flow

A conditional flow is defined as

$$\psi_t(x) = \sigma_t(x_1)x + \mu_t(x_1)$$

Given that σ_t moves from 1 to σ_{min} in a straight line for when t goes from 0 to 1, results in

$$\sigma_t(x_1) = 1 - (1 - \sigma_{min})t$$

The opposite applies to μ_t which moves linearly with t, from 0 to x_1, meaning that

$$\mu_t(x_1) = tx_1$$

This gives the conditional flow as

$$\psi_t(x) = (1 - (1 - \sigma_{\text{min}})t)x + tx_1 \tag{8}$$

9.3 Conditional Vector Field

Given a conditional probability path as per equation 5:

$$\frac{\mathrm{d}\psi_t(x)}{\mathrm{d}t} = u_t(\psi_t(x)|x_1) \tag{5}$$

and the conditional flow defines as:

$$\psi_t(x) = \sigma_t(x_1)x + \mu_t(x_1)$$

With the following derivative with respect to t.

$$\psi_t'(x) = \sigma_t'(x_1)x + \mu_t'(x_1) \tag{12}$$

Due to ψ_t being invertible (and a $\sigma_t(x_1) > 0$) the following inversion $x = \psi_t^{-1}(y)$ can be made. Where equation 5 now can be rewritten as:

$$\psi_t'(\psi_t^{-1}(y)) = u_t(y|x_1), \tag{13}$$

By rewriting the conditional flow equation as follows:

$$x = \frac{\psi_t(x) - \mu_t(x_1)}{\sigma_t(x_1)}.$$

An expression for $\psi_t^{-1}(y)$ can be derived as:

$$\psi_t^{-1}(y) = \frac{y - \mu_t(x_1)}{\sigma_t(x_1)}. \tag{14}$$

By plugging equation 12 and 14 into equation 13, we get the following:

$$\begin{aligned} u_t(y|x_1) &= \psi_t'(\psi_t^{-1}(y)) \\ &= \sigma_t'(x_1)\left(\psi_t^{-1}(y)\right) + \mu_t'(x_1) \\ &= \sigma_t'(x_1)\left(\frac{y - \mu_t(x_1)}{\sigma_t(x_1)}\right) + \mu_t'(x_1) \\ &= \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}(y - \mu_t(x_1)) + \mu_t'(x_1) \end{aligned}$$

Whereby we get the following conditional vectorfield (equation 6):

$$u_t(\psi_t(x)|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}(\psi_t(x) - \mu_t(x_1)) + \mu_t'(x_1) \tag{6}$$

9.4 Conditional Vector Field for Optimal transport

With a defined simple gaussian distribution with $\mu_0(x_1) = 0$ and $\sigma_0(x_1) = 1$ all conditional probability paths $p_t(x|x_1)$ converge at $p(x) = \mathcal{N}(x|0, \mathcal{I})$. Combined with a $\mu_1(x_1) = x_1$ and $\sigma_1(x_1) = \sigma_{\text{min}}$ the OT path is defined as the time-linear path from distribution $p_0(x)$ to $p_1(x)$ thereby resulting in the following two equations:

$$\mu_t(x) = tx_1$$

$$\sigma_t(x) = 1 - (1 - \sigma_{\text{min}})t$$

A simple insertion of these values into the equation for the conditional vector field (equation 6):

$$u_t(\psi_t(x)|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)} \left(\psi_t(x) - \mu_t(x_1)\right) + \mu_t'(x_1) \tag{6}$$

Gives the following equation for $u_t(x|x_1)$, which can be rewritten as follows to arrive at equation 7:

$$\begin{aligned} u_t(\psi_t(x)|x_1) &= \frac{-(1-\sigma_{\text{min}})}{1-(1-\sigma_{\text{min}})t}(\psi_t(x) - tx_1) + x_1 \\ &= -\frac{1-\sigma_{\text{min}}}{1-(1-\sigma_{\text{min}})t}\psi_t(x) + \left(\frac{(1-\sigma_{\text{min}})t + 1 - (1-\sigma_{\text{min}})t}{1-(1-\sigma_{\text{min}})t}\right) x_1 \\ &= -\frac{1-\sigma_{\text{min}}}{1-(1-\sigma_{\text{min}})t}\psi_t(x) + \frac{1}{1-(1-\sigma_{\text{min}})t}x_1 \\ &= \frac{x_1 - (1-\sigma_{\text{min}})}{1-(1-\sigma_{\text{min}})t}\psi_t(x) \end{aligned}$$

9.5 Loss function

The loss function for conditional flow matching is

$$L_{CFM}(\theta) = E_{t,q(x_1),p_t(x|x_1)} \|v_t(x) - u_t(x|x_1)\|^2 \tag{15}$$

Subsequently, the loss function for conditional flow matching with conditional flow is

$$L_{CFM}(\theta) = E_{t,q(x_1),p_t(x_0)} \|v_t(\psi(x)) - u_t(\psi(x)|x_1)\|^2 \tag{16}$$

Given that

$$\psi_t(x) = (1 - (1 - \sigma_{\text{min}})t)x + tx_1 \tag{8}$$

and

$$u_t(\psi_t(x)|x_1) = \frac{x_1 - (1-\sigma_{\text{min}})}{1-(1-\sigma_{\text{min}})t}\psi_t(x) \tag{7}$$

we can derive that

$$
\begin{aligned}
u_t(\psi_t(x)|x_1) &= \frac{x_1 - (1-\sigma_{\text{min}}) * ((1-(1-\sigma_{\text{min}})t)x + tx_1)}{1-(1-\sigma_{\text{min}})t} \\
&= \frac{x_1 - (1-\sigma_{\text{min}}) * (1-(1-\sigma_{\text{min}})t)x - (1-\sigma_{\text{min}})tx_1}{1-(1-\sigma_{\text{min}})t} \\
&= \frac{x_1 * (1-(1-\sigma_{\text{min}})t) - (1-\sigma_{\text{min}}) * (1-(1-\sigma_{\text{min}})t)x)}{1-(1-\sigma_{\text{min}})t} \\
&= \frac{(1-(1-\sigma_{\text{min}})t) * (x_1 - (1-\sigma_{\text{min}})x)}{1-(1-\sigma_{\text{min}})t} \\
&= (x_1 - (1-\sigma_{\text{min}}))x
\end{aligned}
$$

.

Combined with 16 we get

$$
\begin{aligned}
L_{CFM}(\theta) &= E_{t,q(x_1),p_t(x_0)} \|v_t(\psi(x_0)) - u_t(\psi(x_0)|x_1)\|^2 \\
&= E_{t,q(x_1),p_t(x_0)} \|v_t(\psi(x_0)) - (x_1 - (1-\sigma_{\text{min}}))x_0\|^2
\end{aligned}
$$

Appendix C: Explanation of FID Score

The Fréchet Inception Distance (FID) score, see equation 10, assesses the quality of generated images relative to real images based on deep learning feature extraction and statistical distance measurement. Where in general the more number of images the more reliable the score becomes. Calculation of the FID involves the following steps:

1. **Feature Extraction:** Utilizing the InceptionV3 model, a deep neural network pre-trained on the ImageNet dataset, both generated and real images are processed to yield 2048-dimensional feature vectors, capturing essential statistical information.
2. **Statistical Analysis:** For each image set (real and generated), the mean vector (μ) and covariance matrix (σ) of the extracted features are computed. These statistical properties encapsulate the distribution of the images' visual content.
3. **FID Calculation:** The FID score quantifies the similarity between the two multivariate Gaussian distributions defined by the mean vectors and covariance matrices of the real (μ_r, σ_r) and generated (μ_g, σ_g) images.

This formula derives from the Wasserstein distance between two Gaussian distributions, focusing on the squared Euclidean distance between the means and the trace of the sum of the covariances, adjusted by the geometric mean of the covariance matrices. The expression $\text{Tr}(\sigma_r + \sigma_g - 2\sqrt{\sigma_r\sigma_g})$ calculates the trace, sum of the diagonal elements, of the difference between the sum of the covariance matrices and twice their geometric mean. Mathematically, this measures the total variance difference between the two distributions, encapsulating both their spread and the correlation between their features.