Probabilistic AI 2025

DT8122 Project Assignment — Flow Matching

Student: William Lavery, william.lavery@it.uu.se
Date: September 14, 2025

DT8122 Probabilistic Artificial Intelligence, Norwegian University of Science and Technology

Contents

1 General Information

Project report submission for DT8122 - Probabilistic Artificial Intelligence (2025). Additional information has been included in the Appendix 3.1.4. This is a work in progress and not part of the submission (unless the main report does not suffice). Code is available on my github page.

2 Theory Questions

2.1 Question 1

Give a short, high-level description of Flow Matching and its 'optimization objective'.

Answer

Flow Matching (FM) is a framework for training continuous normalising flows (CNFs) or ODE-based generative models. Instead of learning a discrete sequence of denoising steps (as in diffusion models), FM learns a vector field that smoothly transports samples from a simple base distribution (e.g. Gaussian) to a complex data distribution.
The model parametrises a time-dependent velocity field $u_\theta(x, t)$, which defines how a sample should move at each time t. Integrating this velocity field over time gives a continuous path (flow) that transforms base samples into data samples.

Instead of explicitly solving the generative ODE during training, FM uses a matching objective that compares the model's velocity field to the "true" velocity field induced by a chosen interpolation between base and data distributions.
In particular, from [1, p. 3] the FM loss is

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\,(x_0,x_1)\sim(p_0,p_{\text{data}})}\left[\,\|u_\theta(x_t,t) - u(x_t,t)\|^2\,\right],$$

where

- $x_0 = x(t=0) \sim p_0$: sample from the base distribution,
- $x_1 = x(t=1) \sim p_{\text{data}}$: sample from the target distribution,
- x_t: an interpolation between x_0 and x_1 at time t,
- $u(x_t, t)$: the target velocity field that moves x_t along this interpolation.

Thus, the model is trained to imitate the ground-truth field that transports mass from p_0 to p_{data}.

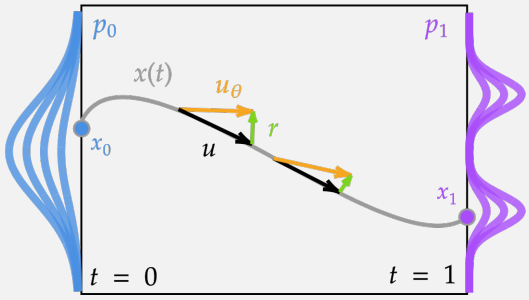

Figure 2.1.1: Trajectory $x(t)$ from $x_0 \sim p_0$ to $x_1 \sim p_1$ according to the true velocity field u. The learned field u_θ (orange), and residual r (green) at two interpolation points between x_0 and x_1 are shown.

In short, FM trains a neural ODE by regressing its velocity field against an analytically defined 'true' flow field, ensuring that when integrated, it maps the base distribution into the data distribution.

More information. See see Appendix 3.3.1 - 3.3.2 for more details on (i) CNFs and there connection to residual flows, and (ii) how FM arises from CNFs. 3.3.2.

2.2 Question 2

Normalizing Flow typically train one network for each layer/transformation, explain briefly how Flow Matching and Contineous Normalizing Flows (CNF) use the same network for every continuous t value.

Answer

This answer closely follows material from [2]. In standard discrete normalising flows (NFs), one constructs a sequence of K invertible transformations

$$\phi_K \circ \phi_{K-1} \circ \cdots \circ \phi_1(x_0),$$

with $x_0 \sim p_0$. Typically, each transformation $\phi_k : \mathbb{R}^d \to \mathbb{R}^d$ is parameterised by a separate neural network (or a block of parameters):

$$\phi_k(x) \approx x + \Delta t\, u_{\theta_k}(x, t_k),$$

with separate parameters θ_k. That is, ϕ_k is designed to look like an Euler discretisation of an ODE step such that at the k-th stage:

$$x_k = \phi_k(x_{k-1}) \approx x_{k-1} + \Delta t\, u_{\theta_k}(x, t_k).$$

Thus, there are K sets of parameters in total leading so parameter count and memory scale linearly with the number of layers.

Both CNFs and FM are continuous-time approaches which replace the discrete composition by the solution of an ODE:

$$\frac{dx(t)}{dt} = u_\theta(x(t), t), \qquad x(0) = x_0.$$

Here, $u_\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is a single vector field, parameterised by one neural network (with parameters θ). Specifically, the letting $\Delta t = \frac{1}{K}$, $K \to \infty$ with a shared parameterization θ, produces

$$x(t + \Delta t) \approx x(t) + \Delta t\, u_\theta(x(t), t).$$

Hence, in the continuous case, there is no need for different networks per time step: the single network $u_\theta(\cdot)$ plays the role of all infinitesimal residual layers simultaneously. The dependence on continuous t is handled by either concatenating t as an input channel to the network, i.e. $u_\theta(x, t)$, or using explicit time embeddings or encodings of t as additional inputs. Thus, the same neural network governs the entire family of transformations $\{\phi_t\}_{t\in[0,T]}$, rather than having a separate network for each time slice.

In CNFs, density dynamics are determined by

$$\frac{d}{dt} \log p(x(t), t) = -\nabla \cdot u_\theta(x(t), t).$$

Both $x(t)$ and $\log p(x(t), t)$ are evolved jointly using the same neural network $u_\theta(\cdot)$ for all t. In particular, no separate "layer" parameters are introduced; time acts as a continuous index controlling the transformation.

In FM training is based on regressing $u_\theta(x, t)$ toward a target vector field $u(x, t)$, which is analytically known for certain interpolation choices (e.g. linear interpolation between base and target). As in CNFs, t is provided as input, so the same parameters generate the dynamics at every instant.

2.3 Question 3

Compare Flow Matching and Continuous Normalizing Flows? Similarities/differences, pros/cons etc. You may structure this discussion as lists or bullet points

Answer

Theory is standard but here primarily based on [2] and [3]. Similarities between FM and CNFs:

- *Common theory.* Both model a flow $x_t \equiv x(t)$ solving the ODE

$$\frac{dx_t}{dt} = u(x_t, t),$$

  The induced densities (ρ_t) satisfy the continuity equation

$$\partial_t \rho_t(x) + \nabla \cdot \big(\rho_t(x)\, u(x,t)\big) = 0,$$

  implying along trajectories the log-density identity

$$\frac{d}{dt} \log \rho_t(x_t) = -\nabla \cdot u(x_t, t).$$

- *Neural vector fields and numerics.* Both parameterize $u_\theta(x,t)$ with time conditioning and effective integrate ODEs for sampling.

Key differences between the two approaches:

- *Objective, supervision, and data.*
  - **CNFs:** maximize likelihood of data. Using the results above at the end-points,

$$\log p_1(x_1) = \log p_0(x_0) - \int_0^1 \nabla \cdot u_\theta(x_t, t)\, dt, \quad x_t = \phi_{0\to t}(x_0),$$

    so the loss is self-supervised via likelihood gradients and requires only samples from p_{data} (and a base p_0).
  - **FM:** regress the learned field to a *prescribed* target field u. One specifies a coupling $\psi(x_0, x_1)$ and an interpolation path $\Gamma(x_0, x_1, t)$; the supervision field is

$$u(x,t) = \mathbb{E}_\psi \big[\partial_t \Gamma(x_0, x_1, t) \,\big|\, \Gamma(x_0, x_1, t) = x\big],$$

    and training minimizes $\mathbb{E}\,\|u_\theta(x,t) - u(x,t)\|^2$.
- *Density evaluation and guarantees.*
  - **CNFs:** provide *tractable, exact* log-densities via the divergence integral, hence genuine normalising flows with calibrated likelihoods.
  - **FM:** do not yield densities by default since the loss avoids $\nabla \cdot u_\theta$. One can estimate $\nabla \cdot u_\theta$ and integrate it to recover likelihoods, but this is not trained-for and may be noisy.
- *Computational profile.*
  - **CNFs:** training requires ODE solvers and divergence estimation at each step, increasing time/memory and variance [2].
  - **FM:** training reduces to pointwise regression at random (x,t) without divergence terms; ODE solvers are deferred to sampling.

Pros and cons are given in Table 2.3.1.

Method	Pros	Cons
CNFs	Exact maximum likelihood training and tractable density evaluation for probabilistic tasks.	Computationally demanding due to Jacobian trace terms.
	Principled: grounded in flows that are both invertible and measure-preserving.	Slower and less stable in high-dimensional problems.
FM	Training is simple and efficient (no log-determinants).	No explicit likelihood or tractable density evaluation.
	Scales to high-dimensional data (e.g. images) with good effect.	Requires construction of intermediate flows between base and data.
	Unifies score-based diffusion and flow-based models.	Theoretical foundations are more heuristic than CNFs.

Table 2.3.1: Pros and cons of CNFs and FM using [2] and [3]. In practice, FM is often favoured for large-scale generative modelling, while CNFs remain valuable when explicit likelihood estimation is required. [1]

2.4 Question 4

Do a similar comparison of Flow Matching and Diffusion.

Answer

Similarities between FM and Diffusion:

- *Time-indexed transport of measures.* Both define a path of marginals $(p_t)_{t\in[0,1]}$ connecting a simple base p_0 (e.g. Gaussian) to $p_1 = p_{\text{data}}$, with sampling obtained by integrating a time-dependent field from t=0 to t=1.
- *Probability flow ODE connection*[3]. Diffusion with forward SDE

$$dx_t = h(x_t, t)\, dt + g(t)\, dW_t$$

admits a reverse-time SDE and an equivalent probability flow ODE

$$\frac{dx_t}{dt} = h(x_t, t) - \tfrac{1}{2} g(t)^2\, \nabla_x \log p_t(x_t).$$

FM directly learns an ODE $\frac{dx_t}{dt} = u_\theta(x_t, t)$; when the FM path matches the diffusion marginals, the optimal u coincides with the probability-flow drift above (see below).

- *Direct relation* [4]. If FM adopts the Gaussian path used by a diffusion model,

$$x_t = \alpha(t)\, x_1 + \sigma(t)\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I),$$

and uses the conditional coupling (x_1, ε), then the regression target reduces to the diffusion probability-flow ODE drift

$$u(x, t) = h(x, t) - \tfrac{1}{2} g(t)^2\, \nabla_x \log p_t(x).$$

Note, $\alpha(t)$ and $\sigma(t)$ define the noise schedule. Hence FM can deterministically reproduce a given diffusion process when their paths match.

- *Neural parameterization & ODE/SDE solvers.* Both use time-conditioned networks and numerical integration at generation; neither requires Jacobian determinants during training.

Key differences between the two approaches:

- *What is learned (target signal).*
  - **Diffusion:** learns the *score* $s_\theta(x,t) \approx \nabla_x \log p_t(x)$ via denoising score matching (DSM) with a known perturbation kernel $q_t(x_t \mid x_1) = \mathcal{N}\big(x_t;\, \alpha(t)\, x_1,\ \sigma(t)^2 I\big)$ using targets

$$\nabla_x \log q_t(x_t|x_1) = -\frac{x_t - \alpha(t)x_1}{\sigma(t)^2}.$$

  - **FM:** learns a *velocity* $u_\theta(x,t)$ by regression to a supervision field (as seen in section 2.3):

$$u(x,t) = \mathbb{E}_\psi\big[\partial_t \Gamma(x_0,x_1,t) \,\big|\, \Gamma(x_0,x_1,t) = x\big]\,.$$

- *Stochastic vs deterministic sampling.*
  - **Diffusion:** default sampler integrates the reverse SDE and so stochasticicty leads outcome diversity.
  - **FM:** sampling is deterministic by construction.
- *Objective and supervision source.*
  - **Diffusion:** purely data-driven; requires only $x_1 \sim p_{\text{data}}$ and the known noising kernel q_t; no access to p_t is needed during training thanks to DSM.
  - **FM:** requires a chosen path & coupling to define (x_t, u) pairs. The quality of u (and thus bias/variance of learning) depends on Γ and the coupling $\pi(x_0, x_1)$.
- *Practical compute profile.*
  - **Diffusion:** training is cheap per step, but high-quality sampling often uses many steps.
  - **FM:** training is pointwise regression on (x,t) pairs; sampling integrates the ODE and can reach good quality in fewer function evaluations when u_θ is smooth.

Method	**Pros**	**Cons**
FM	Simple, stable training as vector-field regression; no Jacobians or score targets; deterministic ODE sampling.	No native likelihoods; density requires post hoc divergence integration
	Flexible path design to inject priors; recovers diffusion's probability-flow ODE when using the same Gaussian path.	Requires a chosen coupling path; supervision bias/variance depends on this choice. Sampling quality and speed hinge on path design.
	Scales well to high-dimensional data	
Diffusion	Strong empirical sample quality; training via denoising score matching is simple and requires only data samples; no Jacobians.	Sampling is costly by default (reverse SDE often needs many number of function evaluations (NFEs)) [5].
	Likelihoods are (in principle) available via the probability-flow ODE divergence integral.	Performance is sensitive to noise schedule and loss reweighting [6].
	Offers stochastic and deterministic samplers.	

Table 2.4.1: Pros and cons of FM and Diffusion. Except for particular citations shown, FM/diffusion portion of the table was formed in the main using [2]/[4].

3 Flow Matching Experiments

In this project, three experimental configurations are explored, each differing in how the source and target pairs are constructed. These are discussed in more detail in Section 3.1. In the first configuration (C1), the source inputs are independent Gaussian noise images, while the targets are clean MNIST digits drawn at random. The second configuration (C2) instead takes the source inputs to be cropped versions of MNIST digits, with the corresponding clean digits as the targets. In this case there is a strong dependency between source and target, as they match everywhere outside the cropped region. However, because pairings are assigned by random shuffling, this natural dependency is not preserved: cropped digits are frequently matched with the wrong clean digits, which disrupts the learning process. Finally, the third configuration (C3) uses the same cropped-noise sources as in C2, but now each source is always paired deterministically with its true clean counterpart.

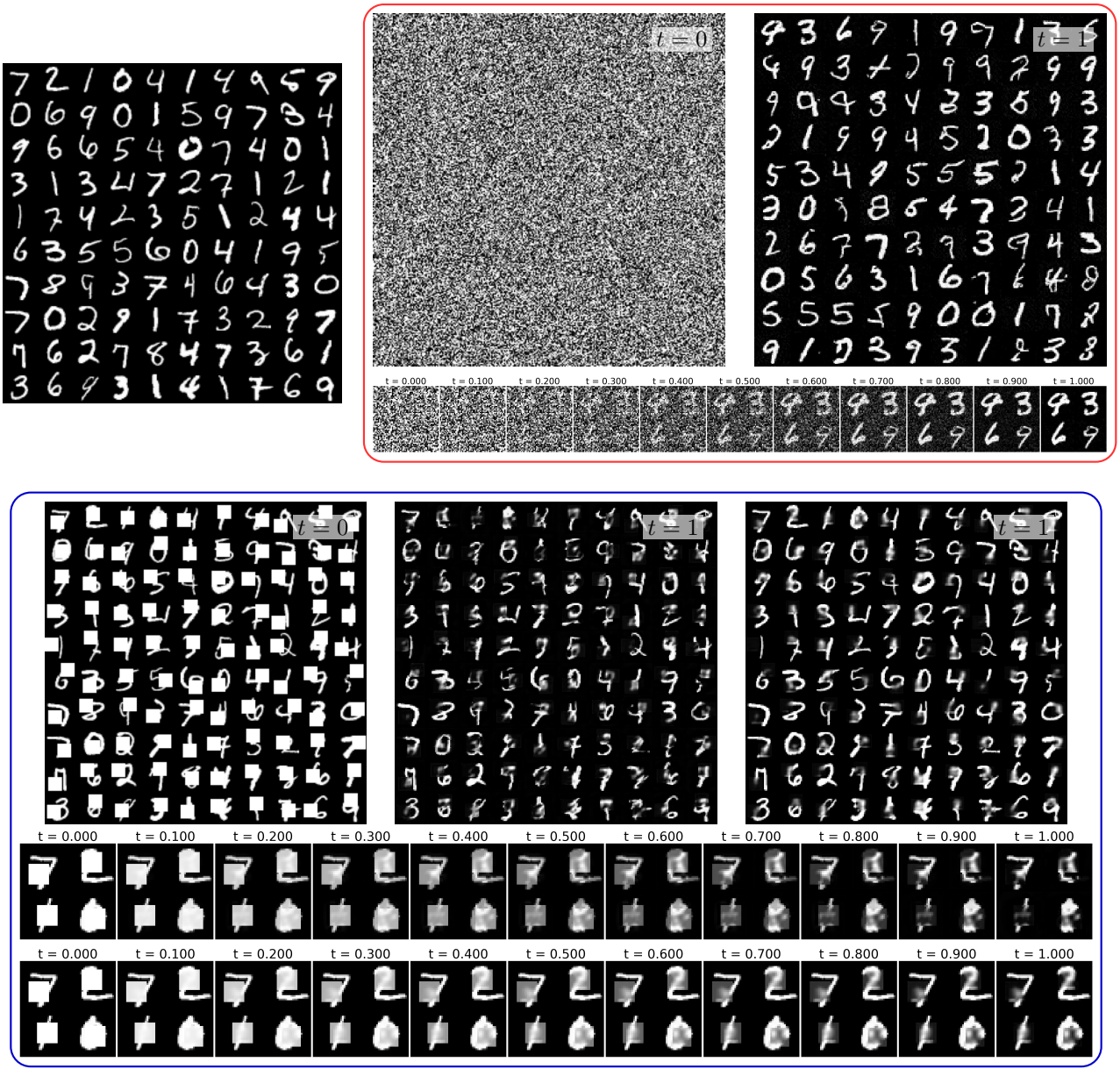

Figure 3.0.1: Flow matching results. The first plot (not surrounded by a box) is the first 100 digits of the ground truth for C2 and C3. C1 is completely ignorant of this. **C1 plots in red box**. First row. 100 examples of generated digits (left) and corresponding model predictions at $t = 0$ (left) and $t = 1$ (right). Second row. 4 examples of generated digits at increasing t. **C2 and C3 plots in blue box**. First row. 100 examples of generated digits at $t = 0$ (left) and $t = 1$ for C2 and C3 (middle and right respectively). Third and fourth rows. 4 examples of generated digits at increasing t values, shown for C2 (row 3) and C3 (row 5). One observes that C3 leads to significantly better recovery of the noise-free digits.

3.1 Discussion

3.1.1 Definitions in the implementation.

In the notation of [7] for a pair consisting of a source sample x_0 and data sample x_1: $z = (x_0, x_1)$ the parameters $q(z)$, $\mu_t(z)$ and σ_t are defined as per Table 3.1.1.

Quantity	**Configuration 1**	**Configuration 2**	**Configuration 3**
Source distribution $q(x_0) = p_0(x_0)$	$x_0 \sim \mathcal{N}(0, I)$, i.i.d. over 28×28 pixels.	x_0 is obtained by cropping a random 12×12 patch of an MNIST digit and replacing it with noise: $x_0 = \alpha(x_1, M) \equiv (1 - M) \odot x_1 + M$, where $\odot$ denotes the element-wise product and binary crop mask $M \in \{0, 1\}^{H \times W}$	
Data distribution $q(x_1) = p_{\text{data}}(x_1)$	MNIST digits normalized to $[-1, 1]$.		
Joint distribution $q(z) = q(x_1, x_2)$	Independent product coupling: $q(z) = q(x_0)q(x_1)$.	Approximately product coupling: x_1 are constructed from x_0 but shuffling essentially breaks the true pairs $q(z) \approx q(x_0)q(x_1)$.	Deterministic coupling: each x_0 is paired with its own clean x_1: $q(x_0, x_1) = \sum_M p_{\text{data}}(x_1)\, p(M)\, \mathbf{1}[x_0 = \alpha(x_1, M)]$.
Interpolation path x_t	For $t \sim \text{Uniform}[0, 1]$, $x_t = (1 - t)\, x_0 + t\, x_1$.		
Conditional $\mu_t(z)$ **and** σ_t	Deterministic linear path $\mu_t(z) = (1 - t)\, x_0 + t\, x_1$, $\sigma_t = 0$.		
Target velocity $u_t(x_t \mid z)$	$u_t(x_t \mid z) = \frac{d}{dt}\mu_t(z) = x_1 - x_0$. This is exactly the regression target used to train u_θ.		

Table 3.1.1: Definitions of the source, data, and coupling distributions, together with path, mean, variance, and target velocity. The three cases correspond to different choices of `cropNoise` and `shufflePairings` in the implementation. Namely, `cropNoise` $= 0/1$ corresponds to Gaussian/crop noise, and `shufflePairings` $= 0/1$ corresponds to randomly shuffling/no shuffling of the the indices of the z pairs before training. In particular, when shuffling is applied each x_0 is matched with a randomly chosen x_1 from a different true pair, resulting in random noise–digit pairings.

3.1.2 Architecture

The model is a U-Net motivated by [1] and implemented myself, but informed by [8, 9]. The network takes as input the current state $x_t \in \mathbb{R}^{1 \times H \times W}$ together with a conditioning on time. The time variable $t \in [0, 1]$ is first embedded using a small Fourier basis together with the raw scalar value. These time features are broadcast across the spatial dimensions and concatenated channel-wise with the grayscale image, producing an input tensor of dimension $(1 + T_{\text{feats}}) \times H \times W$. In this way, every pixel has direct access to the current time.

The encoder follows the standard U-Net pattern: each stage applies two 3×3 convolutions with Group Normalization and SiLU activations (the `DoubleConv` block), followed by average pooling to halve spatial resolution. The number of channels doubles with each downsampling step, beginning with $(1 + T_{\text{feats}}) \to 64$, then $64 \to 128$, and finally $128 \to 256$. At the coarsest level, a bottleneck block further mixes global information.

The decoder mirrors the encoder, progressively upsampling by a factor of two using bilinear interpolation. At each resolution, the upsampled features are concatenated with the corresponding encoder activations via skip connections. Each concatenated tensor is processed by a DoubleConv block. After three such upsampling stages, the decoder produces a 64-channel feature map at the original resolution. A final 3×3 convolution maps this to a single output channel, corresponding to the predicted velocity field $u_\theta(x_t, t)$ at each pixel. The output layer is initialized to zero so that training begins from a near-trivial velocity field.

Training objective. The network is trained with mean-squared error to match the conditional velocity defined by the flow-matching objective. In the deterministic linear case; the target velocity is simply the displacement between clean data and source sample, $u_t = x_1 - x_0$. With a small amount of experimentation with the described architecture and run-time constraints there seemed little benefit to using the stochastic approach, and so the simple linear case was maintained.

3.1.3 Quality comparison

For a fixed number of epochs, C1 yields the best marginal sample quality, C3 is intermediate, and C2 the worst. The cause is primarily the training coupling which fixes what the model must regress, and the resulting conditioning of the problem. In C2 the coupling is unpaired, so for similar intermediate states the supervision

points toward multiple outcomes; with ℓ_2 loss the model averages those modes, washing out the digits. C3 uses the correct pairing, which removes ambiguity but still demands large corrections inside masked regions, keeping variance high. C1's Gaussian-to-data coupling produces smoother, more consistent supervision, yielding a seemingly well-conditioned regression and so sharper samples.

Under a 50-epoch budget, early stopping (no $\geq 1\%$ improvement for 10 consecutive epochs) occurred at C1=17 and C3=33, whereas C2 failed to meet the criterion and would have run to the 50-epoch limit; this ordering mirrors the increasing source to target transport difficulty. Note, Adam optimizer with learning rate of `1e-3` was used throughout.

3.1.4 Optimal transport

In [7], minibatch optimal transport (OT) is defined between two independently drawn batches $\{x_0^i\}_{i=1}^B \sim q_0$ and $\{x_1^j\}_{j=1}^B \sim p_{\text{data}}$. Given a cost function $c(x_0, x_1)$ the OT coupling π minimizes $\sum_{i,j} \pi_{ij}\, c(x_0^i, x_1^j)$, $\pi \in \Pi(\{x_0^i\}, \{x_1^j\})$, where Π denotes the set of couplings with prescribed marginals. This is computed within each minibatch. If a minibatch happens to contain the true clean counterpart of each cropped image, then under ℓ_2 cost $c(x, y) = \|x - y\|_2^2$ OT should pair them, since they differ only within the mask. However, for small minibatches the true pair is rarely in the same minibatch. OT then finds the nearest available x_1 (often a similar digit).

In the implementation, mini-batch OT is not employed; equivalently, the cost matrix $C_{ij} = c(x_0^{(i)}, x_1^{(j)})$ is treated as a constant, so the resulting coupling is determined entirely by the input ordering and pair shuffling.

C1 (Gaussian source, no shuffling). Here $x_0 \sim \mathcal{N}(0, I)$ is independent of $x_1 \sim p_{\text{data}}$, so there is no geometric signal linking the two marginals in pixel space. In this case, using OT brings no systematic benefit: for standard costs, the distances are essentially random with respect to semantic structure, and the induced permutation behaves like a random shuffle. Moreover, computing OT adds nontrivial overhead without improving the coupling. Thus, omitting OT in C1 is reasonable.

C2 (cropped source, shuffled). Now x_0 and x_1 do share structure outside the crop region. However, since pairings are randomly shuffled almost all source samples are matched to semantically mismatched targets leading to identity drift. Significant improvements could be obtained by, for example, introducing hard OT at the mini-batch level and increasing batch size so that, with high probability, each x_0 can find a closely matching x_1 in the same batch. In the limit where the batch spans the entire training set, the hard-OT solution in C2 approaches the deterministic true-pair coupling of C3.

C3 (cropped source, deterministic pairing). Each x_0 is paired with its own clean counterpart x_1 which is the ideal assignment under l_2 costs. In this setting, OT is unnecessary because the optimal pairing is formed a priori.

Summary. In C1, omitting OT avoids unnecessary computation with no loss in coupling quality. In C2, hard mini-batch OT with larger batches yields substantial gains by aligning cropped sources to consistent clean targets within each batch. C3 already realises the optimal coupling, so OT is redundant. The results seen in 3.0.1 are consistent with these theoretical expectations.

References

[1] Yaron Lipman et al. *Flow Matching for Generative Modeling*. 2023. arXiv: 2210.02747 [cs.LG]. URL: https://arxiv.org/abs/2210.02747.

[2] Tor Fjelde, Emile Mathieu, and Vincent Dutordoir. *An Introduction to Flow Matching*. Jan. 2024. URL: https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html.

[3] Anne Gagneux et al. "A Visual Dive into Conditional Flow Matching". In: *ICLR Blogposts 2025*. 2025. URL: https://dl.heeere.com/conditional-flow-matching/blog/conditional-flow-matching/.

[4] Ruiqi Gao and Emiel Hoogeboom and Jonathan Heek and Valentin De Bortoli and Kevin Patrick Murphy and Tim Salimans. *Diffusion Meets Flow Matching: Two Sides of the Same Coin*. Online blog post. https://diffusionflow.github.io, accessed September 9, 2025. 2025.

[5] Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. *Enhancing and Accelerating Diffusion-Based Inverse Problem Solving through Measurements Optimization*. 2024. arXiv: 2412.03941 [cs.CV]. URL: https://arxiv.org/abs/2412.03941.

[6] Zeeshan Patel, James DeLoye, and Lance Mathias. *Exploring Diffusion and Flow Matching Under Generator Matching*. arXiv preprint arXiv:2412.11024v2 [cs.LG]. Version 2, submitted December 17, 2024, licensed under CC BY 4.0. Dec. 2024.

[7] Alexander Tong et al. "Improving and generalizing flow-based generative models with minibatch optimal transport". In: *Transactions on Machine Learning Research* (2024). Expert Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=CD9Snc73AW.

[8] milesial. *Pytorch-UNet: PyTorch implementation of the U-Net for image semantic segmentation*. GitHub repository. Accessed: 2025-09-09. 2025. URL: https://github.com/milesial/Pytorch-UNet.

[9] OpenAI. *guided-diffusion: Diffusion Models for Image Synthesis (GitHub Repository)*. GitHub repository. MIT License; Accessed: 2025-09-09. 2025. URL: https://github.com/openai/guided-diffusion.

Appendix

3.2 Notation

Symbol	Description
$d \in \mathbb{N}$	Data dimension.
t	Continuous time index $t \in [0, T]$; $T > 0$ terminal time (often $T = 1$).
$x \in \mathbb{R}^d$	A point/state in d-dimensional space.
$x(t)$, x_t	State at time t following the flow; shorthand $x_t := x(t)$.
$x_0 := x(0)$	Initial state; $x_0 \sim p_0$.
$x_T := x(T)$	Terminal state; has density p_T.
$\phi_t : \mathbb{R}^d \to \mathbb{R}^d$	Flow map generated by the ODE, $\phi_t(x_0) = x(t)$.
$\phi_{s \to t}$	Flow map from time s to t.
$\Phi^{(}h)_T$	Composition of residual flows up to T.
$u(x, t)$	Target vector field; shorthand $u_t(x) := u(x, t)$.
$u_\theta(x, t)$	Parametric vector field with parameters θ.
$J_u(x, t)$	Jacobian of u w.r.t. x: $J_u(x, t) = \frac{\partial u}{\partial x}(x, t) \in \mathbb{R}^{d \times d}$.
p_{data}	Empirical/true data distribution over $\mathbb{R}^d$.
p_θ	Model distribution induced by u_θ at terminal time T.
ρ_t	Density of x_t at time t for a induced path.
p_t	Density of x_t at time t for a prescribed path.
p_0, p_T	Initial/base density and terminal density along the flow.
$(\phi_{s \to t})_\# p$	Pushforward of p by $\phi_{s \to t}$ (distributional transport).
$\delta(\cdot)$	Dirac measure (used for point-mass boundary conditions).
$\mathcal{L}(\theta)$	Maximum-likelihood objective: $\mathbb{E}_{x \sim p_{\text{data}}}[\log p_\theta(x)]$.
$\mathcal{L}_{\text{FM}}(\theta)$	Flow Matching loss (regression of u_θ to target u_t).
$\mathcal{L}_{\text{pair}}(\theta)$	Paired-regression form.
$\mathcal{L}_{\text{CFM}}(\theta)$	Conditional Flow Matching loss (conditioned FM).
ρ	Sampling distribution over t used inside FM/CFM (e.g., $\mathcal{U}[0, 1]$).
$\mathbb{E}[\cdot]$	Expectation; subscripts indicate sampling law, e.g. $\mathbb{E}_{t \sim \rho}$.
$q_0(\cdot \mid z_0)$, $q_1(\cdot \mid z_1)$	Endpoint laws at $t = 0$ and $t = T$.
X_t	State at time t with $(X_t, Z) \sim p_t(x \mid z)\, p(z)$, hence $X_t \sim p_t$.
U_t	Abbreviation $U_t := u_t(X_t \mid Z)$ (a square-integrable random vector).
$\mathbb{E}_Z[\cdot]$	Shorthand for conditional expectation $\mathbb{E}[\cdot \mid X_t]$.
z	Generic conditioning variable (auxiliary information).
$\lVert \cdot \rVert$	Euclidean norm in $\mathbb{R}^d$.
$\text{tr}(\cdot)$	Matrix trace.

Table 3.2.1: Notation used throughout document.

3.3 Appendix II: Theory

Based closely on [2] and using approaches from [7], the theoretical background leading up to Conditional Flow Matching (CFM) is briefly reviewed. In particular:

1. **Continuous Normalizing Flows (CNFs).** These are introduced as a generative modelling framework based on ordinary differential equations. CNFs can be seen as a precursor to flow matching, and the related variant residual flows are also considered here.
2. **From CNFs to Flow Matching (FM).** Certain pitfalls in the CNF training procedure—such as computational inefficiency and instability—motivate the development of Flow Matching, which provides a more direct and tractable training objective.
3. **Conditional Flow Matching (CFM).** Building on FM, Conditional Flow Matching further addresses limitations by conditioning on auxiliary variables, yielding improved flexibility and training dynamics.

3.3.1 Continuous Normalising Flows

Setup and definition. The state space is $\mathbb{R}^d$. A continuous normalising flow (CNF) is defined by an ordinary differential equation (ODE):

$$\frac{dx(t)}{dt} = u(x(t), t), \qquad x(0) = x_0,$$

where $u : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is a vector field (assumed smooth enough, e.g. Lipschitz in x).

ODE theory and invertibility. From standard ODE existence/uniqueness theorems:

- If u is Lipschitz in x and continuous in t, then for each x_0 there exists a unique solution $t \mapsto x(t)$.
- Define the *flow map*
$$\phi_t : \mathbb{R}^d \to \mathbb{R}^d, \qquad \phi_t(x_0) = x(t).$$
- The uniqueness of solutions implies invertibility: if $\phi_t(x_0) = \phi_t(y_0)$ then $x_0 = y_0$. Moreover, solving the ODE backwards in time (replace t by $-t$) yields the inverse map $\phi_t^{-1} = \phi_{-t}$.

Thus, $\{\phi_t\}_{t\in[0,T]}$ is a smooth family of diffeomorphisms, giving us an invertible transformation of densities. We assume an initial (tractable) density p_0 for $x_0 \sim p_0$, and denote by p_t the density of $x(t) = \phi_t(x_0)$.

Derivation

Infinitesimal change of variables. Consider an infinitesimal step from t to $t+\Delta t$. State update (Euler step):

$$x(t + \Delta t) = x(t) + u(x(t), t)\, \Delta t + o(\Delta t).$$

Jacobian of the update:

$$\frac{\partial x(t + \Delta t)}{\partial x(t)} = I + J_u(x(t), t)\, \Delta t + o(\Delta t), \quad J_u(x, t) = \frac{\partial u}{\partial x}(x, t) \in \mathbb{R}^{d\times d}.$$

Log-determinant expansion. For small ϵ,

$$\det(I + \epsilon A) = 1 + \epsilon\ \mathrm{tr}(A) + o(\epsilon), \quad \log\det(I + \epsilon A) = \epsilon\ \mathrm{tr}(A) + o(\epsilon).$$

Hence,

$$\begin{aligned}
\log\left|\det\left(\tfrac{\partial x(t+\Delta t)}{\partial x(t)}\right)\right| &= \log\left|\det\left(\tfrac{\partial}{\partial x(t)}\{x(t) + u(x(t), t)\, \Delta t + o(\Delta t)\}\right)\right| \\
&= \log\left|\det\left(\{1 + J_u(x(t), t)\, \Delta t + o(\Delta t)\}\right)\right| \\
&= \mathrm{tr}(J_u(x(t), t))\, \Delta t + o(\Delta t).
\end{aligned}$$

Density update (finite-dimensional change of variables). General change of variables gives:

$$p(y) = \frac{p(x)}{|\det\Big(\frac{\partial y}{\partial x}\Big)|}.$$

Short intuition. Let X be a $\mathbb{R}^d$-valued random variable with density p_X, and let $Y = g(X)$ for a smooth, *locally* invertible map $g : \mathbb{R}^d \to \mathbb{R}^d$ with Jacobian matrix $J_g(x) := \frac{\partial g}{\partial x}(x)$. At a point x_0, the map is well-approximated by its linearization:

$$y \approx g(x_0) + J_g(x_0)\,(x - x_0).$$

Under a linear map $y = Ax$ with A invertible, volumes scale by $|\det A|$. Therefore an infinitesimal volume element $\mathrm{d}x$ near x_0 is mapped to a volume element $\mathrm{d}y$ near $y_0 = g(x_0)$ with

$$\mathrm{d}y \approx \big|\det J_g(x_0)\big|\,\mathrm{d}x.$$

Conservation of probability mass in that infinitesimal element, $p_X(x_0)\,\mathrm{d}x \approx p_Y(y_0)\,\mathrm{d}y$, yields the heuristic

$$p_Y\big(g(x_0)\big) \;\approx\; \frac{p_X(x_0)}{\big|\det J_g(x_0)\big|}.$$

Making this precise gives the standard change-of-variables formula. So here,

$$p(x(t+\Delta t), t+\Delta t) = \frac{p(x(t),t)}{\Big|\det\Big(\frac{\partial x(t+\Delta t)}{\partial x(t)}\Big)\Big|}.$$

Taking logs,

$$\log p(x(t+\Delta t), t+\Delta t) = \log p(x(t),t) - \operatorname{tr}(J_u(x(t),t))\,\Delta t - o(\Delta t).$$

Differential equation for the density. Subtracting $\log p(x(t),t)$, dividing by Δt, letting $\Delta t \to 0$ and invoking the continuity equation gives

$$\frac{d}{dt}\log p(x(t),t) = -\operatorname{tr}\Big(J_u(x(t),t)\Big) = -\nabla\cdot u(x(t),t).$$

This is the instantaneous change-of-variables formula. Integrating along a trajectory $x(t) = \phi_t(x_0)$ from $t = 0$ to $t = T$:

$$\log p(x(T),T) - \log p(x(0),0) = -\int_0^T \operatorname{tr}\big(J_u(x(t),t)\big)\,dt.$$

Since $x(0) = x_0$, $x(T) = x_T$:

$$\log p_T(x_T) = \log p_0(x_0) - \int_0^T \nabla\cdot u(x(t),t)\,dt, \quad x(t) = \phi_t(x_0).$$

Practical ODE system. In practice, for a parmeterised vector field neural network: $u_\theta : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$ one integrates jointly:

$$\begin{cases} \dfrac{dx}{dt} = u_\theta(x,t), \\ \dfrac{d}{dt}\log p(x,t) = -\nabla\cdot u_\theta(x,t), \end{cases}$$

with initial conditions $x(0) \sim p_0$, $\log p(x(0),0) = \log p_0(x(0))$. At $t = T$, one obtains both $x(T)$ and its log-density $\log p_T(x(T))$.

Example

Consider a deterministic flow transporting

$$\rho_0 = \mathcal{N}(0,1) \quad \longrightarrow \quad \rho_1 = \mathcal{N}(\mu,1).$$

Define the ODE

$$\frac{dx}{dt} = u(x,t) = \mu, \qquad x(0) \sim \mathcal{N}(0,1).$$

Its solution is

$$x(t) = x(0) + \mu t.$$

Since $x(t)$ is an affine transformation of $x(0)$, and $x(0)$ is Gaussian, $x(t)$ is also Gaussian. In general, if $X \sim \mathcal{N}(m, \sigma^2), \quad Y = aX + b$, then $Y \sim \mathcal{N}(am + b,\, a^2\sigma^2)$. Thus $x(t) \sim \mathcal{N}(\mu t,\, 1)$. Therefore the marginal density at time t is

$$p_t(x) = \mathcal{N}(\mu t,\, 1)$$

This interpolates continuously between $p_0(x) = \mathcal{N}(0,1) = \rho_0$ and $p_1(x) = \mathcal{N}(\mu, 1) = \rho_1$. Note, one can check that u does indeed bridge between ρ_0 and ρ_1 via the continuity equation.

Aside

Residual flow. A depth-N residual network (or residual flow) with step size $h > 0$ applies the iterative update

$$x_{k+1} \;=\; x_k + h\, u(x_k, t_k), \qquad k = 0, \ldots, N-1, \quad t_k = kh,$$

with $Nh = T$. The overall map is the composition

$$\Phi_T^{(h)} \;=\; \big(Id + hu(\cdot, t_{N-1})\big) \circ \cdots \circ \big(Id + hu(\cdot, t_0)\big).$$

Relation to the continuous flow. This is exactly the explicit Euler discretization of the ODE system introduced in Section 3.3.1:

$$\frac{dx(t)}{dt} = u(x(t), t), \qquad x(0) = x_0,$$

whose flow map is ϕ_t. By standard numerical analysis:

$$\Phi_T^{(h)}(x_0) \;\approx\; \phi_T(x_0),$$

with the approximation error vanishing uniformly on $h \to 0$. Thus the residual flow converges to the continuous normalizing flow.

Density transformation. Since each residual block $x \mapsto x + hu(x, t_k)$ is a smooth perturbation of the identity, its Jacobian determinant can be expanded exactly as in the infinitesimal change-of-variables argument of Section 3.3.1. In particular,

$$\log\det\Big(\tfrac{\partial}{\partial x}\big(x + hu(x, t_k)\big)\Big) = h\,\nabla\cdot u(x, t_k) + o(h).$$

Therefore the cumulative density update across layers recovers, in the limit $h \to 0$, the continuous log-density ODE

$$\frac{d}{dt}\log p(x(t), t) = -\,\nabla\cdot u(x(t), t),$$

already derived in Section 3.3.1. At finite h, the residual flow provides a tractable discrete approximation to the same transformation.

Training CNFs. Similarly to other flow-based models CNFs are trained by maximum log-likelihood:

$$\mathcal{L}(\theta) \quad = \mathbb{E}_{x\sim p_{\text{data}}}\big[\log p_\theta(x)\big],$$

where the expectation is taken over the data distribution p_{data} and p_θ is the parametric distribution induced by the CNF. This training procedure involves integrating the time evolution of both the samples x_t and their log-likelihood $\log p_\theta(x_t)$, each governed by the parametric vector field $u_\theta(x, t)$. Concretely, one needs to solve system (**??**). Therefore, training requires (i) expensive numerical ODE simulations at each iteration and (ii) effective estimates of the divergence term $\nabla \cdot u_\theta$ are needed in high dimensions. CNFs are highly expressive, as they parametrize a large class of flows (and thus probability distributions). However, training is typically very

slow because of the ODE integration required for every batch.

3.3.2 Flow Matching

This motivates the FM which provides a simulation free approach to training CNFs by replacing likelihood maximization with a regression objective directly on the parametric vector field u_θ. Specifically, set

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\rho}\,\mathbb{E}_{x_t\sim p_t}\left[\tfrac{1}{2}\,\|u_\theta(x,t) - u(x,t)\|^2\right],$$

where $u(x,t)$ is a target vector field that induces a probability path $\{p_t\}_{t\in[0,1]}$ interpolating between a reference distribution p_0 and the data distribution p_1, i.e.,

$$\log p_1(x) = \log p_0 - \int_0^1 \boldsymbol{\nabla}\cdot u(x,t)dt$$

In words, FM trains u_θ by regressing it toward u_t for all times t. Of course, this assumes access to a suitable u_t. If u_t was already known exactly, there would be no need to learn u_θ at all. However, as seen in the next section, one can cleverly construct useful targets for u_θ without ever having to compute u_t explicitly.

3.3.3 Conditional Flow Matching

Paths of Distributions

A path of distributions is a continuous family of distributions $\{p_t(x)\}_{t\in[0,1]}$ that smoothly interpolates between the two endpoints: $p_0(x)$ (source distribution) and $p_1(x)$ (target distribution).

Analogy: Consider a stadium full of people (particles). At time $t = 0$ the crowd is uniformly spread out (a Gaussian fog). At time $t = 1$, everyone has taken positions to form the shape of a digit (the data distribution). The path $\{p_t\}$ is the 'movie' of how the crowd deforms between these two states. Describing the whole crowd's movement at once can be complicated. Instead, introduce an auxiliary latent variable Z (prior $p(z)$) and define a conditional distribution $p_t(x \mid z)$, which describes the position of a particle at time t given some extra information z. The actual crowd distribution is then the *marginal path*:

$$p_t(x) \;=\; \int p_t(x \mid z)\, p(z)\, dz.$$

that interpolates between p_0 and p_1, i.e.

$$p_0(x) \;=\; \int p_0(x \mid z)\, p(z)\, dz, \qquad p_1(x) \;=\; \int p_1(x \mid z)\, p(z)\, dz.$$

The boundary conditions are

$$p_{t=0}(x \mid z) = q_0(x \mid z), \quad p_{t=1}(x \mid z) = q_1(x \mid z),$$
$$\int q_0(x \mid z)\, p(z)\, dz = p_0(x), \quad \int q_1(x \mid z)\, p(z)\, dz = p_1(x).$$

Example

Conditioning on endpoints. For example, z could be the "assigned exit gate" for a person in the stadium. Draw $z \sim p(z)$ (gate popularity). Given z, the person's initial in-stadium location X_0 (their seat) is drawn from a gate-conditional distribution

$$X_0 \,\big|\, z \;\sim\; q_0(x \mid z),$$

so the overall initial crowd distribution is the mixture $p_0(x) \;=\; \int q_0(x \mid z)\, p(z)\, dz$ (everyone inside at $t = 0$). At the end, each gate z corresponds to an exit location $g(z) \in \mathbb{R}^d$ (point-valued mapping). If we model people as arriving exactly at the gate point at $t = 1$, then

$$X_1 \,\big|\, z \;=\; g(z) \quad\Longrightarrow\quad p_1(x \mid z) \;=\; \delta\big(x - g(z)\big).$$

Marginalizing over gate assignments recovers the final distribution across all exits: $p_1(x) = \int \delta\big(x - g(z)\big)\, p(z)\, dz$ (everyone at their assigned exits at $t = 1$). Thus the boundary conditions read

$$p_{t=0}(x \mid z) = q_0(x \mid z), \qquad p_{t=1}(x \mid z) = \delta\big(x - g(z)\big).$$

For a simple straight-line motion from seats to gates, set

$$X_t \;=\; (1-t)X_0 + t\, g(z), \qquad t \in [0,1],$$

with $X_0 \mid z \sim q_0(\cdot \mid z)$. Then for $t \in [0,1)$ the conditional density is the pushforward

$$p_t(x \mid z) \;=\; \frac{1}{(1-t)^d}\, q_0\!\left(\frac{x - t\, g(z)}{1-t} \,\middle|\, z\right).$$

Two–sided conditioning. Extend the stadium story as follows. Each person is assigned a *start group* z_0 (e.g., a block of seats or fan section) *and* an *exit group* z_1 (e.g., a cluster of gates they tend to use). The pair $(z_0, z_1) \sim p(z_0, z_1)$ captures real correlations: people from certain sections are more likely to head to certain exits.

- Given the start group z_0, the person's exact seat X_0 is drawn from the within-section spread $q_0(\cdot \mid z_0)$: $X_0 \mid z_0 \sim q_0(\,\cdot \mid z_0)$
- Given the exit group z_1, their exact exit point X_1 is drawn from the exit-cluster spread $q_1(\cdot \mid z_1)$ (e.g., near the doors or turnstiles for that group): $X_1 \mid z_1 \sim q_1(\,\cdot \mid z_1)$.

Between $t = 0$ and $t = 1$, imagine a straight walkway that carries each person from their sampled seat to their sampled exit:

$$X_t = (1-t)X_0 + tX_1.$$

At any intermediate time t, the crowd you see near a location x is a *superposition of bridges* from all start–exit pairs: for each (z_0, z_1), you look at all seats likely under $q_0(\cdot \mid z_0)$ that would, when linearly blended with exits likely under $q_1(\cdot \mid z_1)$, place people at x at time t. This yields the conditional density

$$\begin{aligned} p_t(x \mid z_0, z_1) &= \iint \delta\big(x - (1-t)x_0 - tx_1\big)\, q_0(x_0 \mid z_0)\, q_1(x_1 \mid z_1)\, dx_0 dx_1, \\ &= \frac{1}{(1-t)^d} \int q_0\!\left(\frac{x - tx_1}{1-t} \,\middle|\, z_0\right)\, q_1(x_1 \mid z_1)\, dx_1. \end{aligned}$$

i.e., the pushforward of the *pair* of endpoint spreads through the linear bridge.
Marginally (ignoring individual assignments), you mix over all start–exit pair types weighted by their popularity $p(z_0, z_1)$:

$$p_t(x) = \iint p_t(x \mid z_0, z_1)\, p(z_0, z_1)\, dz_0 dz_1,$$

so the observed crowd is a blend of many "section→exit" flows. The endpoints are preserved: at $t = 0$ you recover the mixture of seat sections $p_0(x) = \int q_0(x \mid z_0)\, p_0(z_0)\, dz_0$, and at $t = 1$ you recover the mixture of exit clusters $p_1(x) = \int q_1(x \mid z_1)\, p_1(z_1)\, dz_1$:

$$\begin{aligned} p_0(x) &= \iint q_0(x \mid z_0)\, p(z_0, z_1)\, dz_0 dz_1 = \int q_0(x \mid z_0)\, p_0(z_0)\, dz_0, \\ p_1(x) &= \iint q_1(x \mid z_1)\, p(z_0, z_1)\, dz_0 dz_1 = \int q_1(x \mid z_1)\, p_1(z_1)\, dz_1, \end{aligned}$$

where $p_0(z_0) = \int p(z_0, z_1)\, dz_1$ and $p_1(z_1) = \int p(z_0, z_1)\, dz_0$.

Transport Equation Given a latent z (e.g. the assigned exit gate of a person), the conditional distribution $p_t(x \mid z)$ evolves in time $t \in [0,1]$ according to a transport equation driven by a *conditional vector field* $u_t(x \mid z)$:

$$\partial_t p_t(x \mid z) \;+\; \nabla_x \cdot \big(p_t(x \mid z)\, u_t(x \mid z)\big) \;=\; 0.$$

Derivation

Let Z be a latent variable with prior $p(z)$, and let

$$p_t(x \mid z), \qquad t \in [0,1],\ x \in \mathbb{R}^d,$$

denote a family of *conditional* probability densities (the "sub-crowd" with gate z), with *marginal* (unconditional) density

$$p_t(x) \;=\; \int p_t(x \mid z)\, p(z)\, dz.$$

Assume that for each fixed z, the *conditional dynamics* of particles is governed by the ODE

$$\frac{d}{dt} X_t^{(z)} \;=\; u_t\big(X_t^{(z)} \mid z\big), \qquad X_0^{(z)} \sim q_0(\cdot \mid z), \tag{1}$$

for a (sufficiently regular) *conditional vector field* $u_t(\cdot \mid z)$. Fix z and consider a smooth compactly-supported test function $\varphi : \mathbb{R}^d \to \mathbb{R}$. By the chain rule and (1),

$$\frac{d}{dt}\, \mathbb{E}\big[\varphi(X_t^{(z)})\big] \;=\; \mathbb{E}\Big[\nabla\varphi\big(X_t^{(z)}\big) \cdot u_t\big(X_t^{(z)} \mid z\big)\Big].$$

Writing this in terms of the conditional density $p_t(\cdot \mid z)$,

$$\frac{d}{dt} \int \varphi(x)\, p_t(x \mid z)\, dx \;=\; \int \nabla\varphi(x) \cdot u_t(x \mid z)\; p_t(x \mid z)\, dx.$$

An integration by parts (no boundary term for compact support) yields the *weak form*

$$\int \varphi(x)\, \partial_t p_t(x \mid z)\, dx \;=\; -\int \varphi(x)\, \nabla_x \cdot \big(p_t(x \mid z)\, u_t(x \mid z)\big)\, dx.$$

Since this holds for all such φ, the *conditional continuity (transport) equation* is obtained:

$$\partial_t p_t(x \mid z) \;+\; \nabla_x \cdot \big(p_t(x \mid z)\, u_t(x \mid z)\big) \;=\; 0 \quad \text{(in the distributional sense).} \tag{2}$$

Interpretation (analogy): for each gate z, the sub-crowd density $p_t(\cdot \mid z)$ is carried along by the velocity plan $u_t(\cdot \mid z)$; probability mass is conserved along the flow.

Lipma et al. (2023) introduced the notion of CFM by noticing that this conditional vector field can express the marginal vector field of interest via the conditional probability path. This is now shown. Differentiate the marginal density and use (2):

$$\begin{aligned}
\partial_t p_t(x) = \partial_t \int p_t(x \mid z)\, p(z)\, dz \;&=\; \int \partial_t p_t(x \mid z)\, p(z)\, dz \\
&= -\int \nabla_x \cdot \big(p_t(x \mid z)\, u_t(x \mid z)\big)\, p(z)\, dz = -\,\nabla_x \cdot \left(\int p_t(x \mid z)\, u_t(x \mid z)\, p(z)\, dz \right)
\end{aligned}$$

By definition, a *marginal* velocity field $u_t(x)$ for p_t is any vector field satisfying

$$\partial_t p_t(x) \;+\; \nabla_x \cdot \big(p_t(x)\, u_t(x)\big) \;=\; 0.$$

Comparing with the previous line, a sufficient choice (unique wherever $p_t(x) > 0$) is

$$p_t(x)\, u_t(x) \;=\; \int p_t(x \mid z)\, u_t(x \mid z)\, p(z)\, dz \quad \Longrightarrow \quad u_t(x) \;=\; \frac{1}{p_t(x)} \int p_t(x \mid z)\, u_t(x \mid z)\, p(z)\, dz. \tag{3}$$

From this formulation one can use $\mathcal{L}_{\mathrm{CFM}}(\theta)$ instead of $\mathcal{L}_{\mathrm{FM}}(\theta)$ to train the parametric vector field u_θ .

Derivation

Assume a (teacher) conditional velocity $u_t(x \mid z)$ transports $p_t(\cdot \mid z)$:

$$\partial_t p_t(x \mid z) + \nabla_x \cdot \big(p_t(x \mid z)\, u_t(x \mid z)\big) = 0.$$

Define the *marginal velocity* $u_t(x)$ as in (3):

$$u_t(x) = \frac{1}{p_t(x)} \int p_t(x \mid z)\, u_t(x \mid z)\, p(z)\, dz = \mathbb{E}\big[u_t(x \mid Z) \,\big|\, X_t = x\big] = \mathbb{E}[u_t(X_t \mid Z)]\,,$$

where $X_t \sim p_t$ drawn from $p_t(x \mid z)p(z)$. Let $u_\theta(x, t)$ be the model (unconditional in z for now); assume t is drawn from some density $\rho(t)$ independent of (X_t, Z). The (time-weighted) FM objective is

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\rho}\, \mathbb{E}_{X_t\sim p_t}\Big[\tfrac{1}{2}\, \big\|u_\theta(X_t, t) - u_t(X_t)\big\|^2\Big]\,. \tag{4}$$

Let X be any random variable, U a square–integrable random vector, and $a(X)$ any measurable function. Then

$$\mathbb{E}\big[\|a(X) - \mathbb{E}[U \mid X]\|^2\big] = \mathbb{E}\big[\mathbb{E}\big[\|a(X) - U\|^2 \,\big|\, X\big]\big] - \mathbb{E}[\mathrm{Var}(U \mid X)]\,. \tag{5}$$

Fix $t \in [0, 1]$ and define

$$U_t := u_t(X_t \mid Z), \qquad (X_t, Z) \sim p_t(x \mid z)p(z).$$

Then $\mathbb{E}[U_t \mid X_t] = u_t(X_t)$. Apply (5) with $X \leftarrow X_t$, $U \leftarrow U_t$, $a(X) \leftarrow u_\theta(X_t, t)$:

$$\begin{aligned}\mathbb{E}_{X_t}\Big[\big\|u_\theta(X_t, t) - u_t(X_t)\big\|^2\Big] &= \mathbb{E}_{X_t}\Big[\mathbb{E}\Big[\big\|u_\theta(X_t, t) - U_t\big\|^2 \,\Big|\, X_t\Big]\Big] \\ &\quad - \mathbb{E}_{X_t}[\mathrm{Var}(U_t \mid X_t)]\,.\end{aligned} \tag{6}$$

Substituting (6) into (4) and using the shorthand $\mathbb{E}_Z[\cdot]$ for $\mathbb{E}[\cdot|X_t]$ with $(X_t, Z) \sim p_t(x \mid z)p(z)$

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \underbrace{\mathbb{E}_{t\sim\rho}\, \mathbb{E}_{X_t, Z}\big[\tfrac{1}{2}\, \|u_\theta(X_t, t) - u_t(X_t \mid Z)\|^2\big]}_{\mathcal{L}_{\mathrm{pair}}(\theta)} - \underbrace{\tfrac{1}{2}\, \mathbb{E}_{t\sim\rho}\, \mathbb{E}_{X_t}\big[\,\mathrm{Var}(U_t \mid X_t)\big]}_{\text{independent of } \theta}\,. \tag{7}$$

The first term $\mathcal{L}_{\mathrm{pair}}$ is the familiar paired regression objective and is equal to the the CFM loss:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t\sim\rho}\, \mathbb{E}_{X_t, Z}\big[\tfrac{1}{2}\, \|u_\theta(X_t, t) - u_t(X_t \mid Z)\|^2\big]$$

The second term does not depend on θ and thus $\nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta)$.

—