Computational Mathematics and AI

Lecture 6: PDE Framework for Generative Modeling

Lars Ruthotto
Departments of Mathematics and Computer Science
lruthotto@emory.edu
larsruthotto

slido.com **#CBMS25**

Reading List

Historical Context: Generative modeling has evolved from statistical density estimation and graphical models through variational methods and adversarial training to modern diffusion- and flow-based approaches.

Key Readings:

1. Chen et al. (2018) – Neural Ordinary Differential Equations. *NeurIPS*

   Continuous-time neural networks foundation.
2. Benamou and Brenier (2000) – Computational Fluid Mechanics Solution to Monge-Kantorovich. *Numer. Math.*

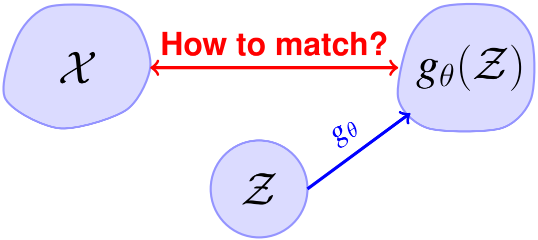
   Dynamic optimal transport formulation.
3. Onken et al. (2021) – Optimal Transport Regularization for Continuous Normalizing Flows. *NeurIPS*

   Penalizing kinetic energy of trajectories to obtain uniqueness and improve efficiency.
4. Lipman et al. (2023) – Flow Matching for Generative Modeling. *ICLR*

   Supervised learning objective for continuous normalizing flows.
5. Song et al. (2021) – Score-Based Generative Modeling via SDEs. *ICLR*
   Unifying framework for diffusion via Fokker-Planck.

Lecture Outline: CNF $\rightarrow$ OT $\rightarrow$ Flow Matching $\rightarrow$ Score-Based Diffusion

Generative Modeling as Distribution Matching



Mathematical Framework

- **Goal**: Learn generator $g_\theta : \mathbb{R}^q \to \mathbb{R}^n$ that transforms latent $\mathcal{Z}$ to match data $\mathcal{X}$
- **Challenges**:
  - n typically large (high-dimensional)
  - $\mathcal{X}$ complicated (multimodal, disjoint support)
- **Core Problem**: Match distributions

$$p_{g_\theta(\mathcal{Z})}(x) \approx p_{\mathcal{X}}(x)$$

- **Today's focus**: Distribution Matching with PDEs

generative modeling = matching high-dimensional distributions

Classical Generative Approaches

Normalizing Flows

- Invertible mapping $x = g_\theta(z)$ with tractable change of variables
- Challenge: Log-determinant expensive, architectural constraints

$$\log p_\theta(x) = \log p_z(g_\theta^{-1}(x)) + \log|\det J_{g_\theta^{-1}}(x)|$$

Variational Autoencoders (VAEs)

- Latent variable model with encoder-decoder structure
- Challenge: Blurry samples, mode collapse

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\mu(z|x)}[\log p(x|z)] - D_{\text{KL}}(q_\mu(z|x)\|p(z))$$

Generative Adversarial Networks (GANs)

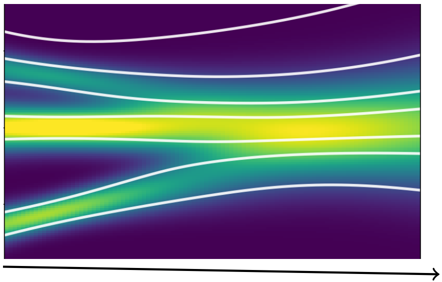
- Adversarial min-max game between generator g_θ and discriminator d_μ
- Challenge: Training instability, no tractable density

$$\min_{g_\theta} \max_{d_\mu} \mathbb{E}_x[\log d_\mu(x)] + \mathbb{E}_z[\log(1 - d_\mu(g_\theta(z)))]$$

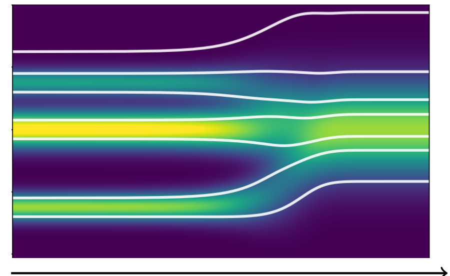
today we focus on continuous-time models with PDEs

Today: PDE Perspective of Generative Modeling

$$\frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t v_t) = 0, \quad \rho_0 = p_X$$



$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v_t) = \frac{g^2(t)}{2}\Delta p_t, \quad p_0 = p_X$$



Roadmap

1. **Continuous Normalizing Flows**: Method of characteristics
2. **Optimal Transport**: Penalize transport costs to accelerate training/sampling
3. **Flow matching**: Even faster training by avoiding time integration
4. **Diffusion**: Stochastic alternative via Fokker-Planck

central theme: derive state-of-the-art generative AI models via simple PDEs

Continuous Normalizing Flows

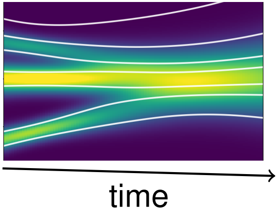
From Continuity Equation to Method of Characteristics

$$\frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t v_t) = 0, \quad t \in (0, 1], \quad \rho_0 = p_{\mathcal{X}}$$

Method of Characteristics

- Define **characteristic curves** (particle trajectories):

$$\frac{dx}{dt} = v_t(x, t), \quad x(0) \sim p_{\mathcal{X}}$$



- Along these curves (log density evolution):

$$\frac{d \log \rho_t(x(t))}{dt} = -\nabla \cdot v_t$$

Key Insight

- **PDE** (continuity equation) $\Longleftrightarrow$ **System of ODEs** (particle trajectories)
- If particles follow ODEs, density automatically satisfies PDE!

CNF Idea: Parameterize velocity field v_t as neural network $v_\theta(z, t)$

PDE transport $\Rightarrow$ Method of characteristic $\Rightarrow$ Neural ODE

CNF Training

Neural ODE

- Velocity field: $\frac{dx}{dt} = v_\theta(x(t), t)$ with $x(0) \sim p_0$
- **Advantage**: Invertible and tractable log-density for any reasonable v_θ
- Density p_t satisfies continuity equation automatically (method of characteristics)

Likelihood Computation (integrate instantaneous change of variables from Slide 5)

$$\log p_\theta(x) = \log p_0(x(1)) + \int_0^1 \nabla \cdot v_\theta(x(t), t) dt$$

Training

- Maximize $\mathbb{E}_{x \sim p_{\mathcal{X}}}[\log p_\theta(x)]$ (maximum likelihood)
- **Requirements**: ODE solve + trace computation at every training step

Sampling: Draw $z(1) \sim p_{\mathcal{Z}}$, solve ODE from $t = 1$ to $t = 0$ with $v_\theta(z, t)$

elegant theory, but ODE solving + trace at every step

CNF Limitations

Computational Bottlenecks

1. **Trace computation**: $O(n)$ per evaluation (or high-variance stochastic)
2. **ODE solving**: Many function evaluations needed for adaptive time-stepping
3. **Training time**: Significantly slower than standard architectures

CNF Problem is Under-Determined

- Only map matters, no control over **trajectory shape**
- Maximum likelihood $\neq$ minimize path energy
- Can produce **complex, curved, high-energy paths**
- More function evaluations needed in training and sampling

Scale Challenge

- Doesn't scale well to high-dimensional imaging applications
- Both trace and ODE bottlenecks compound

Using CNFs for Optimal Transport

Regularizing CNF with Optimal Transport

Idea: Add kinetic energy penalty to MLE loss with trade-off parameter α

$$\mathcal{L}_{\text{OT-CNF}}(\theta) = \mathbb{E}_{x \sim p_{\mathcal{X}}}[-\log p_\theta(x)] + \frac{\alpha}{2}\mathbb{E}\left[\int_0^1 \|v_\theta(x(t), t)\|^2 dt\right]$$

Variational Perspective via Benamou-Brenier

$$\text{minimize} \quad \mathcal{L}(\rho_t, v_t) = \int_0^1 \int \frac{1}{2}\|v_t(x)\|^2 \rho_t(x)\, dx\, dt + \lambda D(\rho_1, p_{\mathcal{Z}})$$

$$\text{subject to} \quad \frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t v_t) = 0, \quad \rho_0 = p_{\mathcal{X}}$$

Structure from Transport Costs (when optimal)

- Optimality condition: $v_t = -\nabla \Phi_t$ (conservative), $\nabla \cdot v_t = -\Delta \Phi_t$
- Value function Φ_t satisfies Hamilton-Jacobi-Bellman equation
- **Key insight**: Transport cost $\rightarrow$ structure $\rightarrow$ simplified computation
- **Note**: CNF can be extended to mean field games

OT-Flow - Learning the Value Function

Idea: Directly learn the value function Φ to exploit gradient structure

Training Objective (simplified from OT-Flow formulation)
Given samples $x_1, \ldots, x_N \sim p_{\mathcal{X}}$, learn $\Phi_\theta(x, t)$ such that:

- Velocity: $v_\theta = -\nabla_x \Phi_\theta$ (conservative by construction)
- Maximize likelihood w.r.t. standard normal $p_1 = \mathcal{N}(0, I)$
- Add penalty terms to enforce Hamilton-Jacobi-Bellman equation

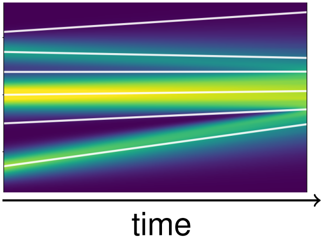
Key Computational Advantage

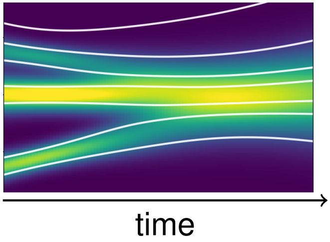
- Divergence: $\nabla \cdot v = -\Delta \Phi_\theta$ (Laplacian)
- Can compute $\Delta \Phi_\theta$ directly with $O(m^2 n)$ operations (where m = network width)
- Enables efficient likelihood computation during training

What OT-Flow Achieves

- **Theoretical**: Unique solution, straighter paths (minimal kinetic energy)
- **Computational**: Explicit Laplacian, fewer function evaluations
- **Sampling**: Can use fewer time steps than vanilla CNF

OT-Flow - Benefits and Limitations





Benefits Over Vanilla CNF

Theoretical:

- Unique solution (OT map)
- Min kinetic energy $\rightarrow$ straighter paths

Computational:

- Explicit Laplacian: $\nabla \cdot v = -\Delta\Phi$
- More sampling efficient
- Fewer steps than vanilla CNF

Why Not Used for Imaging?

MLE Framework Limitations:

- time integration in training
- Likelihood computation: expensive and unstable (manifold hypothesis)

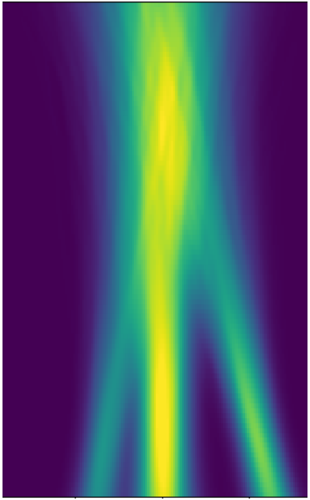
Next: Flow matching and diffusion

- no time integration in training
- simpler, faster training, SOTA sample quality

Flow Matching

Key Idea: Feasible Paths via Superposition

$$\frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t v_t) = 0, \quad \rho_0 = p_{\mathcal{X}}, \quad \rho_1 = p_{\mathcal{Z}}$$



Special Case: Two Dirac Deltas

For point pair $p_{\mathcal{X}} = \delta(x_0)$ and $p_{\mathcal{Z}} = \delta(x_1)$, OT map is

- Conditional path: $\psi_t(x_0, x_1) = (1-t)x_0 + tx_1$
- Conditional density: $\rho_t(\cdot|x_0, x_1) = \delta(x - \psi_t(x_0, x_1))$
- Conditional velocity: $u_t(x|x_0, x_1) = \frac{d\psi_t}{dt} = x_1 - x_0$

Superposition via Linearity

Sample $x_0 \sim p_{\mathcal{X}}$ and $x_1 \sim p_{\mathcal{Z}}$ independently.
By linearity of the PDE, the marginal density is:

$$\rho_t(x) = \int \delta(x - \psi_t(x_0, x_1)) p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1)\, dx_0\, dx_1 = \mathbb{E}_{x_0, x_1}[\rho_t(x|x_0, x_1)]$$

This gives a **feasible** probability path!

Question: How do we get the marginal velocity v_t?

From Conditional to Marginal Velocity

For each (x_0, x_1) the conditional density and conditional velocity satisfy

$$\frac{\partial \rho_t(x|x_0,x_1)}{\partial t} + \nabla \cdot (\rho_t(x|x_0,x_1) u_t(x|x_0,x_1)) = 0$$

Step 1: Take expectation over $(x_0, x_1) \sim p_{\mathcal{X}} \times p_{\mathcal{Z}}$

$$\int \frac{\partial \rho_t(x|x_0,x_1)}{\partial t} p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1) dx_0 dx_1 + \int \nabla \cdot (\rho_t(x|x_0,x_1) u_t(x|x_0,x_1)) p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1) dx_0 dx_1 = 0$$

Step 2: Interchange differentiation and integration

$$\frac{\partial}{\partial t}\left[\int \rho_t(x|x_0,x_1) p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1) dx_0 dx_1\right] + \nabla \cdot \left[\int \rho_t(x|x_0,x_1) u_t(x|x_0,x_1) p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1) dx_0 dx_1\right] = 0$$

Step 3: Identify Coefficients. This is $\frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t v_t) = 0$ with:

$$v_t(x) = \frac{\int u_t(x|x_0,x_1) \rho_t(x|x_0,x_1) p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1) dx_0 dx_1}{\int \rho_t(x|x_0,x_1) p_{\mathcal{X}}(x_0) p_{\mathcal{Z}}(x_1) dx_0 dx_1}$$

marginal velocity is weighted average of conditional velocities

Conditional Expectation Interpretation

From Integral to Conditional Expectation

The marginal velocity can be written as:

$$v_t(x) = \frac{\int u_t(x|x_0, x_1)\rho_t(x|x_0, x_1)p(x_0, x_1)dx_0dx_1}{\int \rho_t(x|x_0, x_1)p(x_0, x_1)dx_0dx_1} = \mathbb{E}[u_t(x|x_0, x_1) \mid \psi_t(x_0, x_1) = x]$$

where $p(x_0, x_1) = p_{\mathcal{X}}(x_1)\mathcal{N}(x_0)$

Problem: Computing this conditional expectation in high dimensions is **intractable**!

Reminder: Conditional Expectation

General form: $\mathbb{E}[Y \mid X = x] = \frac{\int y\, p(y|x)dy}{\int p(y|x)dy}$
Here: $Y = u_t(x|x_0, x_1)$, condition on $\psi_t(x_0, x_1) = x$

next: How to compute this expectation?

From Expectation to Function Approximation

Idea: Regression to Compute Expectation Use a neural network $v_\theta(x,t)$ and minimize:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_0,x_1}[\|v_\theta(\psi_t(x_0,x_1),t) - u_t(x|x_0,x_1)\|^2]$$

where $u_t(x|x_0,x_1) = x_1 - x_0$ is known analytically!

Why This Works: A Simple Example

Two data points with the same x but different y values: (x,y_1) and (x,y_2)

Minimize: $L(v) = |v(x) - y_1|^2 + |v(x) - y_2|^2$

Optimality: $\frac{\partial L}{\partial v(x)} = 2(v(x) - y_1) + 2(v(x) - y_2) = 0 \quad \Rightarrow \quad v^*(x) = \frac{y_1+y_2}{2}$

Key insight: $v^*(x)$ is the **average** of y-values at x = conditional expectation!

Conditional Flow Matching Training

Training Objective

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_0,x_1}[\|v_\theta(\psi_t(x_0,x_1),t) - u_t(x|x_0,x_1)\|^2]$$

Training Procedure

1. Sample $x_0 \sim p_\mathcal{X}$, $x_1 \sim p_\mathcal{Z}$, $t \sim U[0,1]$
2. Compute path location: $\psi_t = (1-t)x_0 + tx_1$
3. Compute target velocity: $u_t = x_1 - x_0$
4. Compute prediction: $v_\theta(\psi_t, t)$
5. Minimize squared error with stochastic gradient descent

Advantages

- No ODE solve during training
- No trace computation
- Simple supervised learning (not adversarial or variational)

Sampling: Solve ODE with learned v_θ from $t = 1$ to $t = 0$ (same as in CNF)

supervised learning on analytically known conditional velocities

Discussion - Flow Matching

The Flow Matching Recipe

1. Construct conditional flows from point pairs (Dirac deltas)
2. Use superposition via linearity $\rightarrow$ marginal densities
3. Fit neural network to match conditional velocities (supervised learning)

Computational Advantages

- **Training**: No ODE solves, no trace estimation $\rightarrow$ significantly faster than CNF
- **Sampling**: Linear interpolation $\rightarrow$ fewer function evaluations
- **Simplicity**: Convexity in v_t, supervised learning

Trade-offs

- Produces **feasible** pairs (satisfies continuity equation) ✓
- Does NOT minimize Benamou-Brenier kinetic energy (not optimal)
- Construction beats optimization in high dimensions!

State-of-the-Art: Stable Diffusion 3, Sora, AlphaFold 3

next: What about stochastic alternatives? (diffusion via Fokker-Planck)

Score-Based Diffusion

Stochastic Alternative - Fokker-Planck PDE

Idea: Use second-order PDE to map data to Gaussian

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot (v_t p_t) + \frac{g^2(t)}{2} \nabla \cdot (\nabla p_t), \quad t > 0, \quad p_0 = p_{\mathcal{X}}$$

Choose v_t so that $p_T(x)$ converges to tractable distribution as $T \to \infty$ (i.e., Gaussian).

Example: Variance-Preserving Formulation (Mean Reversal)

$$v_t(x) = -\frac{1}{2}\beta(t)x, \quad g^2(t) = \beta(t)$$

This gives variance-preserving dynamics: $p_T \to \mathcal{N}(0, I)$

Common choices for $\beta(t)$:

- **Linear:** $\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \cdot t/T$
- **Cosine:** $\beta(t)$ derived from $\alpha_t = \cos(\pi t/2T)$

Our code: $\beta_{\min} = 0.1$, $\beta_{\max} = 20$, $T = 5$

diffusion term causes asymptotic convergence to Gaussian

Log Transform Reveals Score Function

The Score Function

$$s_t(x) := \nabla_x \log p_t(x) = \frac{\nabla p_t(x)}{p_t(x)}$$

Substituting Score into FP-PDE

Recall divergence-gradient form from previous slide. Use $\nabla p_t = p_t s_t$ to rewrite:

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot \left[v_t p_t - \frac{g^2(t)}{2} \nabla p_t \right]$$

Score Matching - Conditional Construction

Idea (similar to flow matching): Start simple and use linearity of PDE!

Construction Strategy

Step 1: Pick $x_0 \sim p_{\mathcal{X}}$, solve FP-PDE with Dirac initial condition $p_0(x) = \delta(x - x_0)$

Solution is Gaussian (from variance-preserving SDE):

$$p_t(x|x_0) = \mathcal{N}(x; \alpha_t x_0, \sigma_t^2 I)$$

where α_t, σ_t are known analytically for the variance-preserving schedule

Step 2: Conditional score (differentiate log pdf of Gaussian)
For $x = \alpha_t x_0 + \sigma_t \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$:

$$s_t(x|x_0) = \nabla_x \log p_t(x|x_0) = -\frac{x - \alpha_t x_0}{\sigma_t^2} = -\frac{\epsilon}{\sigma_t}$$

Next Step: Marginalize

- Similar to flow matching: $p_t(x) = \int p_t(x|x_0) p_{\mathcal{X}}(x_0)\, dx_0 = \mathbb{E}_{x_0}[p_t(x|x_0)]$
- New problem: Computing score is more difficult because log is nonlinear!

next: how to get marginal score from conditional scores

From Conditional to Marginal Score

Marginal density from superposition: $p_t(x) = \int p_t(x|x_0)p_{\mathcal{X}}(x_0)\,dx_0$

Computing the Score (note: cannot simply average!)

Chain rule for positive functions a_i: $\nabla \log \sum_i a_i = \frac{\sum_i \nabla a_i}{\sum_i a_i}$

Apply to marginal:

$$s_t(x) = \nabla_x \log p_t(x) = \frac{\nabla_x \int p_t(x|x_0)p_{\mathcal{X}}(x_0)\,dx_0}{\int p_t(x|x_0)p_{\mathcal{X}}(x_0)\,dx_0}$$

Score-Based Diffusion Training

Training Objective

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x_0,\epsilon}[\|s_\theta(x_t, t) - s_t(x_t|x_0)\|^2]$$

where $s_t(x_t|x_0) = -\frac{\epsilon}{\sigma_t}$ (known analytically!)

Training Procedure

1. Sample $x_0 \sim p_\mathcal{X}$ (data), $\epsilon \sim \mathcal{N}(0, I)$, $t \sim U[0, T]$
2. Compute forward diffusion: $x_t = \alpha_t x_0 + \sigma_t \epsilon$
3. Compute target score: $s_t(x_t|x_0) = -\frac{\epsilon}{\sigma_t}$
4. Compute prediction: $s_\theta(x_t, t)$
5. Minimize squared error with gradient descent

Advantages

- No ODE solve during training
- No trace computation
- Trivial forward process (just add noise)

supervised learning on analytically known conditional scores

Time Reversal and Sampling

Reverse SDE (stochastic, white trajectories)

$$dx = [f(x,t) - g^2(t)s_\theta(x,t)]\,dt + g(t)\,d\bar{W}$$

Time-reversed SDE: samples from $p_0 = p_{\mathcal{X}}$ starting from $p_T \approx \mathcal{N}(0, I)$.

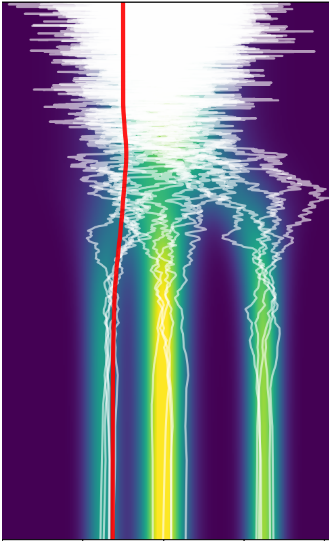
Probability Flow ODE (deterministic, red trajectory)

$$\frac{dx}{dt} = f(x,t) - \frac{g^2(t)}{2} s_\theta(x,t)$$

Same marginals as SDE, but deterministic.

Advantages of ODE

- Faster sampling (adaptive step sizes)
- Exact likelihood computation
- Latent space interpolation



Training and Sampling Characteristics

Training Advantages

- No ODE solving ✓
- No trace computation ✓
- Forward diffusion is trivial (just add noise)
- **Extremely stable** - regression on Gaussian noise

Sampling

- More steps than flow matching (100-1000 vs 20-100 NFE)
- But very high sample quality (SOTA FID scores)
- Extremely robust across different architectures

Trade-off

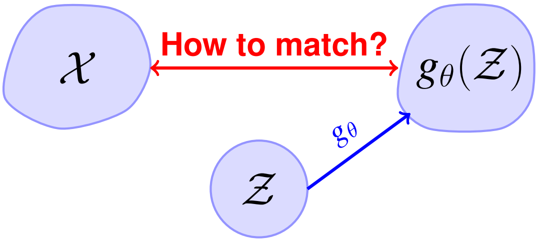
- Slower sampling, but simpler training
- Stochastic spreading vs deterministic transport
- Entropy-regularized OT interpretation (Schrödinger bridge)

State-of-the-Art: DALL-E 2, Imagen, Stable Diffusion (1-2)

extreme training stability, SOTA quality, more sampling steps

Summary

PDE Framework for Generative Modeling



Continuity Equation:

$$\frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t v_t) = 0$$

$$\rho_0 = p_{\mathcal{X}}, \quad \rho_1 = p_{\mathcal{Z}}$$

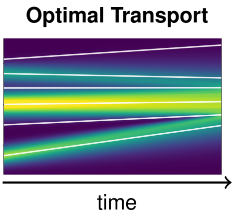
Fokker-Planck Equation:

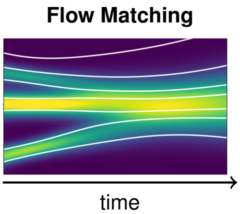
$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v_t) = \frac{g^2(t)}{2} \Delta p_t$$

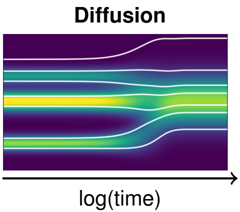
$$p_0 = p_{\mathcal{X}}, \quad p_T \to \mathcal{N}(0, I)$$

More to learn: VAEs, GANs, other approaches

Σ: PDE Approaches for Generative Modeling







Method	**PDE**	**Optimal?**	**Train**	**Sample**	**Key Differentiator**
CNF	Continuity	No	Slow	Slow	MLE with trace bottleneck
OT Flow	Continuity	Yes	Medium	**Fast**	Theory: $v = -\nabla \Phi$
Flow Match	Continuity	No	Fast	Fast	no time integration, arbitrary $p_{\mathcal{Z}}$
Diffusion	Fokker-P	No	Fast	Slow	no time integration, SDE sampling

Key Insights

- All satisfy transport PDEs (feasible), but only OT Flow is optimal
- **Holy grail**: Matching-type algorithms for the actual OT problem (active research)
- Tradeoff: Computational feasibility vs. theoretical optimality

References I

- Benamou, J.-D. and Y. Brenier (2000). "A Computational Fluid Mechanics Solution to the Monge-Kantorovich Mass Transfer Problem". In: *Numerische Mathematik* 84.3, pp. 375–393.
- Chen, R. T. Q. et al. (2018). "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31.
- Lipman, Y. et al. (2023). "Flow Matching for Generative Modeling". In: *International Conference on Learning Representations (ICLR)*.
- Onken, Derek et al. (2021). "OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.10, pp. 9223–9232.
- Song, Y. et al. (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations (ICLR)*.