Flow Matching

The Philosophy Behind Flow Matching

- noise: x0 ~ p0 ### usually gaussian noise
  target: x1 ~ p1 ### whatever target we define; in this case, let's say different kitty pictures
- We believe that certain types of pictures have certain corresponding patterns, from which we can recognize them as "kitty" pictures rather than puppy pictures. These patterns can be represented, clustered, and learned in a high-dimensional space.
- We also believe that, "to generate kitty pictures" is equal to learning the target distribution p1, or learning the high-dimensional structure behind all those kitty pictures.
- Here is the interesting part: we could try to directly learn the target distribution p1. But in flow matching, we would rather learn how to move samples from gaussian noise x0 ~ p0 to the target distribution x1 ~ p1.
- The reason is that we have countless gaussian noise samples. Once we learn the map/function/velocity/flow field from noise to data, we can generate as many pictures as we want. It is also because we have to start from SOMETHING rather than from a "NONE".
- btw, standard gaussian noise is not just meaningless "nothing". It is a simple, well-defined, and easy-to-sample distribution. So don't be misled by the word "noise" in its name.

The Philosophy Behind Flow Matching

- We should be able to imagine a pile of points x0 flowing to another pile of points x1. More precisely, we want the whole distribution p0 to flow into the target distribution p1.
- When training, we pick a point from noise x0 and a point from data x1, then construct a path between them. Along this path, we train the model to predict the velocity that moves x0 toward x1. This is the basic meaning of the FM loss.
- A simple example is:

  $x_t = (1- t)*x0 + t*x1$

- The corresponding velocity is:

  $velocity = x1 - x0$

- So the model learns:

  $v(t, x_t) \approx x1 - x0$

The Philosophy Behind Flow Matching

Instinctively, let's imagine what the ideal flow would look like. ### disagreement/divergence is allowed here

- 1. **The target distribution p1 must be accurate and expressive enough, so that newly generated points can flow to the region that really belongs to the kitty distribution.** In other words, “kitty be kitty”. To achieve this, the latent / data space must be informative enough.
- 2. **One point from x0 should flow to its target smoothly, and its neighbours should also flow to nearby targets with similar velocity vectors.** To achieve this, we may need a better way to pair x0 and x1, instead of using totally random pairings.
- 3. **The velocity field should be locally continuous.** In other words, stability. If two points are close to each other in space and time, their velocities should not suddenly become completely different, unless the data structure really requires that.
- 4. **The path should be short, in a sense.** Shorter or straighter paths are usually easier to integrate and may require fewer sampling steps.
- 5. **Conflicting paths should be avoided.** More precisely, if different training paths pass through similar regions but give very different velocity directions, the model may learn an averaged velocity. This can make the final flow curved, inefficient, or unstable.
- 6. **Fewer steps are better.**

FLOW MATCHING FOR GENERATIVE MODELING

Yaron Lipman[1,2] **Ricky T. Q. Chen**[1] **Heli Ben-Hamu**[2] **Maximilian Nickel**[1] **Matt Le**[1]
[1]Meta AI (FAIR) [2]Weizmann Institute of Science

Preliminaries: CNF

2 PRELIMINARIES: CONTINUOUS NORMALIZING FLOWS

Let $\mathbb{R}^d$ denote the data space with data points $x = (x^1, \ldots, x^d) \in \mathbb{R}^d$. Two important objects we use in this paper are: the *probability density path* $p : [0, 1] \times \mathbb{R}^d \to \mathbb{R}_{>0}$, which is a time dependent[1] probability density function, *i.e.*, $\int p_t(x)dx = 1$, and a *time-dependent vector field*, $v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$. A vector field v_t can be used to construct a time-dependent diffeomorphic map, called a *flow*, $\phi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ordinary differential equation (ODE):

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

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

Previously, Chen et al. (2018) suggested modeling the vector field v_t with a neural network, $v_t(x; \theta)$, where $\theta \in \mathbb{R}^p$ are its learnable parameters, which in turn leads to a deep parametric model of the flow ϕ_t, called a *Continuous Normalizing Flow* (CNF). A CNF is used to reshape a simple prior density p_0 (*e.g.*, pure noise) to a more complicated one, p_1, via the push-forward equation

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

where the push-forward (or change of variables) operator $*$ is defined by

$$[\phi_t]_* p_0(x) = p_0(\phi_t^{-1}(x)) \det \left[\frac{\partial \phi_t^{-1}}{\partial x}(x) \right]. \tag{4}$$

A vector field v_t is said to *generate* a probability density path p_t if its flow ϕ_t satisfies equation 3. One practical way to test if a vector field generates a probability path is using the continuity equation, which is a key component in our proofs, see Appendix A. We recap more information on CNFs, in particular how to compute the probability $p_1(x)$ at an arbitrary point $x \in \mathbb{R}^d$ in Appendix C.

Method

- Target: $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p_t(x)}\|v_t(x) - u_t(x)\|^2,$ regresses ut with a neural network vt
- However, ut is intractable:

$$u_t(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)p_1(x_1)}{p_t(x)}dx_1$$

- We shift the perspective from points along the path to x0 and x1.

$$x_t = (1-t)x_0 + tx_1$$

- Conditional Path:

$$u_t(x_t|x_0, x_1) = x_1 - x_0$$

- Conditional Velocity:

$$\nabla_\theta \mathcal{L}_{FM}(\theta) = \nabla_\theta \mathcal{L}_{CFM}(\theta)$$

- The authors prove that

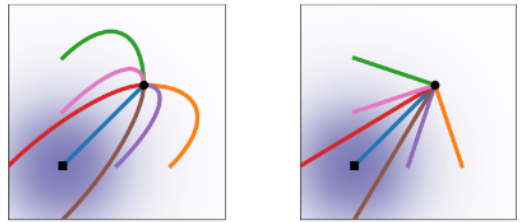


Figure 3: Diffusion and OT conditional trajectories.

What’s the problems now?

- Only meet #1 and #4

Rectified Flow

FLOW STRAIGHT AND FAST: LEARNING TO GENERATE AND TRANSFER DATA WITH RECTIFIED FLOW

Xingchao Liu*, **Chengyue Gong***, **Qiang Liu**
Department of Computer Science
University of Texas at Austin
{xcliu, cygong, lqiang}@cs.utexas.edu

Flow straight and fast: Learning to generate and transfer data with rectified flow
X Liu, C Gong, Q Liu - arXiv preprint arXiv:2209.03003, 2022 - arxiv.org
We present rectified flow, a surprisingly simple approach to learning (neural) ordinary differential equation (ODE) models to transport between two empirically observed ...
☆ Save 99 Cite Cited by 3498 Related articles »

Rectified Flow: flow straight and fast

$$x_t = (1-t)x_0 + tx_1$$

$$v = x_1 - x_0$$

$$\min_v \int_0^1 \mathbb{E}\left[\|(X_1 - X_0) - v(X_t,\, t)\|^2\right] \mathrm{d}t, \quad \text{with} \quad X_t = tX_1 + (1-t)X_0,$$

Rectified Flow : reflow

Step 1: Random Pairing (1-RF)
Train an initial model by randomly pairing noise x0 with real data x1, which results in tangled and highly curved generation trajectories.
Step 2: Deterministic Simulation
Use this initial model to simulate the full forward process, generating a specific synthetic target z1 uniquely driven by the starting noise z0.
Step 3: Causal Re-pairing
Form a new training dataset by discarding the random targets and strictly linking each initial noise z0 to its own deterministically generated output z1.

Step 4: Retraining (2-RF)
Train a new network on these causally linked pairs to directly predict the straight-line velocity vector (v = z1 - z0).
Step 5: Straightened Trajectories
This re-pairing eliminates intersecting paths and physically straightens the ODE trajectories, enabling high-quality generation in a single Euler step.

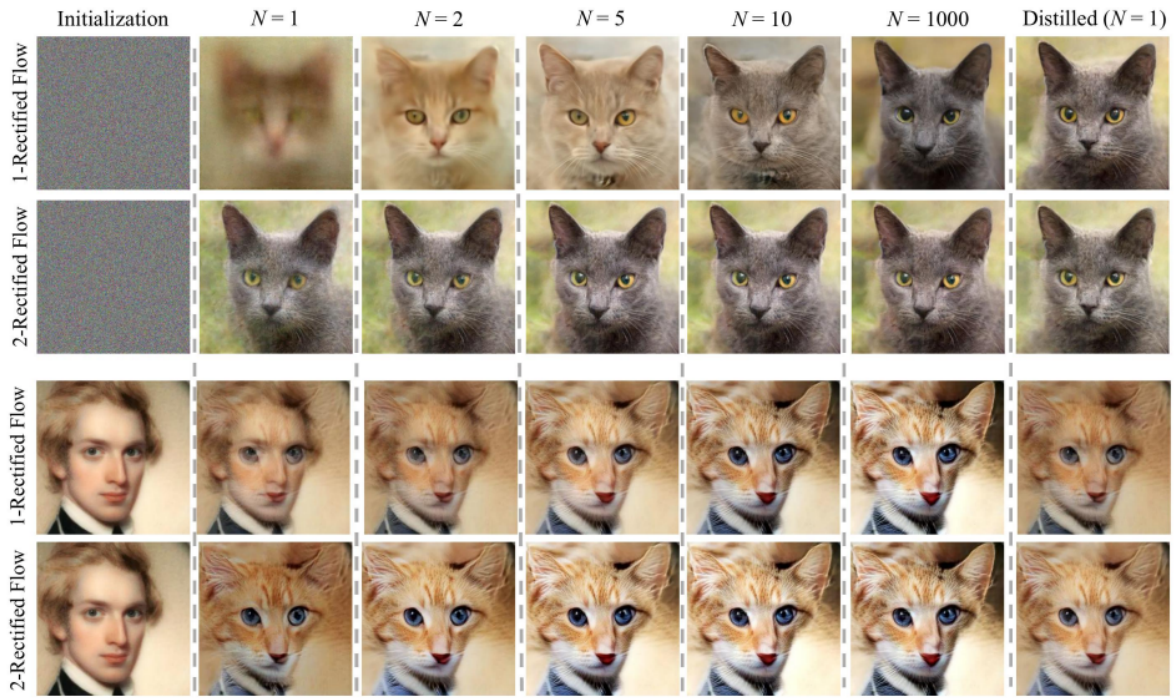


Figure 1: The trajectories of rectified flows for image generation (π_0: standard Gaussian noise, π_1: cat faces, top two rows), and image transfer between human and cat faces (π_0: human faces, π_1: cat faces, bottom two rows), when simulated using Euler method with step size $1/N$ for N steps. The first rectified flow induced from the training data (*1-rectified flow*) yields good results with a very small number (e.g., ≥ 2) of steps; the straightened reflow induced from *1-rectified flow* (denoted as *2-rectified flow*) has nearly straight line trajectories and yield good results even with one discretization step.

OT-CFM

Improving and generalizing flow-based generative models with minibatch optimal transport

A Tong, K Fatras, N Malkin, G Huguet, Y Zhang, J Rector-Brooks, G Wolf, Y Bengio

arXiv preprint arXiv:2302.00482, 2023 · arxiv.org

> Continuous normalizing flows (CNFs) are an attractive generative modeling technique, but they have been held back by limitations in their simulation-based maximum likelihood training. We introduce the generalized conditional flow matching (CFM) technique, a family of simulation-free training objectives for CNFs. CFM features a stable regression objective like that used to train the stochastic flow in diffusion models but enjoys the efficient inference of deterministic flow models. In contrast to both diffusion models and

SHOW MORE

☆ Save 99 Cite Cited by 976 Related articles All 5 versions »

Published in Transactions on Machine Learning Research (03/2024)

Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport

Alexander Tong* *alexander.tong@mila.quebec*
Mila – Québec AI Institute, Université de Montréal

Kilian Fatras* *kilian.fatras@mila.quebec*
Mila – Québec AI Institute, McGill University

Nikolay Malkin* *nikolay.malkin@mila.quebec*
Mila – Québec AI Institute, Université de Montréal

Guillaume Huguet *guillaume.huguet@mila.quebec*
Mila – Québec AI Institute, Université de Montréal

Yanlei Zhang *yanlei.zhang@mila.quebec*
Mila – Québec AI Institute, Université de Montréal

Jarrid Rector-Brooks *jarrid.rector-brooks@mila.quebec*
Mila – Québec AI Institute, Université de Montréal

Guy Wolf *guy.wolf@umontreal.ca*
Mila – Québec AI Institute, Université de Montréal
Canada CIFAR AI Chair

Yoshua Bengio *yoshua.bengio@mila.quebec*
Mila – Québec AI Institute, Université de Montréal
CIFAR Senior Fellow

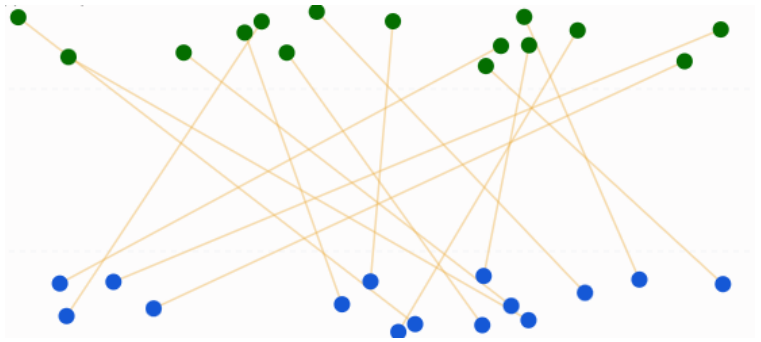
Reviewed on OpenReview: *https://openreview.net/forum?id=CD9Snc73AW*

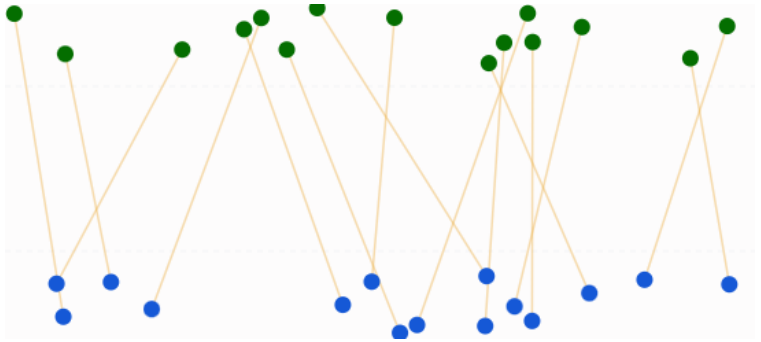
OT-CFM

1. Sample BatchNumber(let’s say 256) x0 and 256 x1.

2. Calculate the MSE between each x0 and x1.

3. Pair each of them so that the MSE is minimum.





Mean Flow

Mean flows for one-step generative modeling

Z Geng, M Deng, X Bai, Z Kolter, K He

Advances in Neural Information Processing Systems, 2026 • proceedings.neurips.cc

Abstract

We propose a principled and effective framework for one-step generative modeling. We introduce the notion of average velocity to characterize flow fields, in contrast to instantaneous velocity modeled by Flow Matching methods. A well-defined identity between average and instantaneous velocities is derived and used to guide neural network training. Our method, termed the\textit {MeanFlow} model, is self-contained and requires no pre-training, distillation, or curriculum learning. MeanFlow demonstrates

SHOW MORE

Save Cite Cited by 423 Related articles All 3 versions

Mean Flows for One-step Generative Modeling

Zhengyang Geng[1]* **Mingyang Deng**[2] **Xingjian Bai**[2] **J. Zico Kolter**[1] **Kaiming He**[2]

[1]CMU [2]MIT

Mean Flow

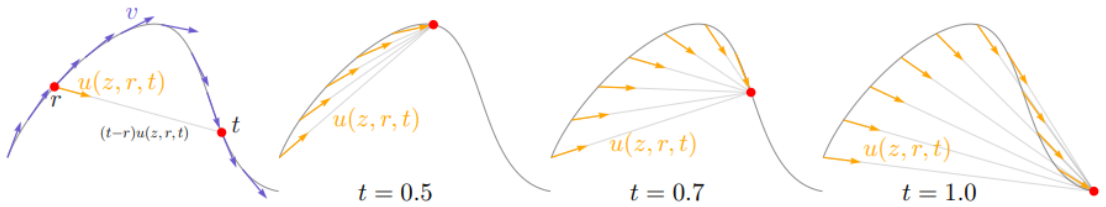


Figure 3: **The field of *average velocity*** $u(z, r, t)$. **Leftmost**: While the *instantaneous velocity* v determines the tangent direction of the path, the average velocity $u(z, r, t)$, defined in Eq. (3), is generally not aligned with v. The average velocity is aligned with the *displacement*, which is $(t - r)u(z, r, t)$. **Right three subplots**: The field $u(z, r, t)$ is conditioned on both r and t, and is shown here for $t = 0.5$, 0.7, and 1.0.

- Motivation: If we want one-step(few steps) generation, we can learn the average velocity over the whole interval instead of the instantaneous velocity.

- Method:

  Average Velocity: $u(z_t, r, t) \triangleq \frac{1}{t-r}\int_r^t v(z_\tau, \tau)d\tau.$

  Again, to avoid integration, we can differentiate both sides.

  Average Velocity $u_\theta(z_t, t) + t\frac{d}{dt}u_\theta(z_t, t) = v(z_t, t) = z_1 - z_0$ (u is learned by NN, du is calculated by Jacobian-vector product (JVP))

  ```
  import jax
  import jax.numpy as jnp
  ```

  One-step generation $z_0 = z_1 - u(z_1, 0, 1)$

What’s the problem now?

- 1. training target involves JVP, which can be numerically unstable
- 2. one-step map can amplify local perturbations ### stability
- 3. endpoint accuracy becomes extremely important

Stable Mean Flow

Stable Mean Flow: Lyapunov-Inspired One-Step Flow Matching

G Zhang, M Haberle, D Geiger

Proceedings of the IEEE/CVF Conference on Computer Vision and ..., 2026 · openaccess.thecvf.com

Abstract

The Mean Flow Matching algorithm is the state-of-the-art for one-step generative models. Building on this idea, we propose the Stable Mean Flow algorithm and introduce a Lyapunov-inspired stability regularizer that enforces local non-expansivity of the single-step transport map. This design guarantees uniqueness of characteristics and bounds trajectory drift. We conduct experiments that show improved output quality and convergence speed over Mean Flow. Moreover, we establish explicit upper bounds on

SHOW MORE

Save Cite Related articles

Stable Mean Flow: Lyapunov-Inspired One-Step Flow Matching

Guangxun Zhang Mason Haberle Davi Geiger

New York University

gz2260@nyu.edu mason.haberle@nyu.edu dg1@nyu.edu

Stable Mean Flow

Definition 1 (Lyapunov Stability). *Let $x(t; t_0, x_0)$ denote the solution of an ODE with initial condition $x(t_0) = x_0$, defined for all $t \geq t_0$. A reference solution $x^*(t) := x(t; t_0, \bar{x}_0)$ is* Lyapunov stable *if, for every $\varepsilon > 0$, there exists $\delta > 0$ such that*

$$\|x_0 - \bar{x}_0\| < \delta \implies \|x(t; t_0, x_0) - x^*(t)\| < \varepsilon \quad \forall t \geq t_0.$$

- Motication: One-step generation can amplify errors (small perturbations, velocity errors, or numerical errors)

- Lyapunov Stability:if two trajectories start sufficiently close, then they should remain close during the subsequent evolution. ### small perturbations remain small

- Method: Philosophy

Local non–expansivity and soft penalty. During the training, we impose *non-expansivity* (NE) softly via the hinge–squared penalty

$$\ell_{\text{stab}}(\Delta z) = \Big[\max\{0,\ \|\Delta z - (t-r)\,\Delta u\|_2 - \|\Delta z\|_2\}\Big]^2,$$

where $\Delta u := u_\theta(z_t + \Delta z, r, t) - u_\theta(z_t, r, t)$ and $\alpha := t - r > 0$. The loss ℓ_{stab} is zero whenever (NE) holds and increases quadratically otherwise. Our final objective is a weighted combination of the Mean Flow loss and the stability loss, $L = \mathcal{L} + \mu\ell_{\text{stab}}$. By choosing proper hyperpa-

Meaning: