Mean Flows for One-step Generative Modeling

一步生成建模的均值流

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

耿正阳 [1*] 邓明阳 [2] 白星剑 [2] J. Zico Kolter [1] 何凯明 [2]

[1] CMU [2] MIT

[1] 卡内基梅隆大学 [2] 麻省理工学院

Abstract

摘要

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 MeanFlow model, is self-contained and requires no pre-training, distillation, or curriculum learning. MeanFlow demonstrates strong empirical performance: it achieves an FID of 3.43 with a single function evaluation (1-NFE) on ImageNet 256×256 trained from scratch, significantly outperforming previous state-of-the-art one-step diffusion/flow models. Our study substantially narrows the gap between one-step diffusion/flow models and their multi-step predecessors, and we hope it will motivate future research to revisit the foundations of these powerful models.

我们提出了一个有原则且高效的一步生成建模框架。我们引入了平均速度的概念来刻画流场，与 Flow Matching 方法中建模的瞬时速度形成对比。推导出平均速度与瞬时速度之间的明确定义恒等式，并用以指导神经网络训练。我们的方法称为 MeanFlow 模型，具有自包含性，无需预训练、蒸馏或课程学习。MeanFlow 在实证上表现优异: 在 ImageNet 256×256 上从零开始训练，单次函数评估 (1-NFE) 即可达到 3.43 的 FID，显著优于之前最先进的一步扩散/流模型。我们的研究大幅缩小了一步扩散/流模型与其多步前辈之间的差距，期望激励未来研究重新审视这些强大模型的理论基础。

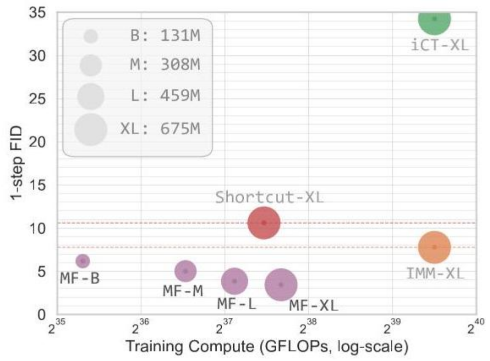


Figure 1: One-step generation on ImageNet 256×256 from scratch. Our MeanFlow (MF) model achieves significantly better generation quality than previous state-of-the-art one-step diffusion/flow methods. Here, iCT [43], Shortcut [13], and our MF are all 1-NFE generation, while IMM's 1-step result [52] involves 2-NFE guidance. Detailed numbers are in Tab. 2. Images shown are generated by our 1-NFE model.

图 1:ImageNet 256×256 上的一步生成，从零开始。我们的 MeanFlow(MF) 模型在生成质量上显著优于之前最先进的一步扩散/流方法。这里，iCT [43]、Shortcut [13] 和我们的 MF 均为 1-NFE 生成，而 IMM 的一步结果 [52] 涉及 2-NFE 引导。详细数据见表 2。展示的图像均由我们的 1-NFE 模型生成。

1 Introduction

1 引言

The goal of generative modeling is to transform a prior distribution into the data distribution. Flow Matching [28, 2, 30] provides an intuitive and conceptually simple framework for constructing flow paths that transport one distribution to another. Closely related to diffusion models [42, 44, 19], Flow Matching focuses on the velocity fields that guide model training. Since its introduction, Flow Matching has seen widespread adoption in modern generative modeling [11, 33, 35].

生成建模的目标是将先验分布转换为数据分布。Flow Matching [28, 2, 30] 提供了一个直观且概念简单的框架, 用于构建将一个分布传输到另一个分布的流路径。与扩散模型 [42, 44, 19] 密切相关, Flow Matching 聚焦于指导模型训练的速度场。自引入以来，Flow Matching 已被广泛应用于现代生成建模 [11, 33, 35]。

Both Flow Matching and diffusion models perform iterative sampling during generation. Recent research has paid significant attention to few-step—and in particular, one-step, feedforward—generative models. Pioneering this direction, Consistency Models [46, 43, 15, 31] introduce a consistency constraint to network outputs for inputs sampled along the same path. Despite encouraging results, the consistency constraint is imposed as a property of the network's behavior, while the properties of the underlying ground-truth field that should guide learning remain unknown. Consequently, training can be unstable and requires a carefully designed "discretization curriculum" [46, 43, 15] to progressively constrain the time domain. In this work, we propose a principled and effective framework, termed MeanFlow, for one-step generation. The core idea is to introduce a new ground-truth field representing the average velocity, in contrast to the instantaneous velocity typically modeled in Flow Matching. Average velocity is defined as the ratio of displacement to a time interval, with displacement given by the time integral of the instantaneous velocity. Solely originated from this definition, we derive a well-defined, intrinsic relation between the average and instantaneous velocities, which naturally serves as a principled basis for guiding network training.

Flow Matching 和扩散模型在生成过程中均执行迭代采样。近期研究高度关注少步数，尤其是一步前馈生成模型。作为该方向的先驱，Consistency Models [46, 43, 15, 31] 引入了一致性约束，要求网络输出对沿同一路径采样的输入保持一致。尽管结果令人鼓舞，但一致性约束作为网络行为的性质被强加，而应指导学习的底层真实场的性质仍未知。因此，训练可能不稳定，需精心设计“离散化课程”[46, 43, 15] 以逐步约束时间域。在本工作中，我们提出了一个有原则且高效的一步生成框架，称为 MeanFlow。核心思想是引入一个新的真实场，表示平均速度，与 Flow Matching 中通常建模的瞬时速度形成对比。平均速度定义为位移与时间间隔的比值，位移由瞬时速度的时间积分给出。仅基于此定义，我们推导出平均速度与瞬时速度之间的明确定义的内在关系，自然成为指导网络训练的有原则基础。

Building on this fundamental concept, we train a neural network to directly model the average velocity field. We introduce a loss function that encourages the network to satisfy the intrinsic relation between average and instantaneous velocities. No extra consistency heuristic is needed. The existence of the ground-truth target field ensures that the optimal solution is, in principle, independent of the specific network, which in practice can lead to more robust and stable training. We further show that our framework can naturally incorporate classifier-free guidance (CFG) [18] into the target field, incurring no additional cost at sampling time when guidance is used.

基于这一基本概念，我们训练神经网络直接建模平均速度场。我们引入了一个损失函数，鼓励网络满足平均速度与瞬时速度之间的内在关系。无需额外的一致性启发式。真实目标场的存在确保最优解原则上与具体网络无关，实际中可带来更稳健和稳定的训练。我们进一步展示了该框架可自然地将无分类器引导 (CFG)[18] 整合进目标场，在使用引导时采样阶段无额外开销。

Our MeanFlow Models demonstrate strong empirical performance in one-step generative modeling. On ImageNet 256 $\times$256 [7], our method achieves an FID of 3.43 using 1-NFE (Number of Function Evaluations) generation. This result significantly outperforms previous state-of-the-art methods in its class by a relative margin of 50% to 70% (Fig. 1). In addition, our method stands as a self-contained generative model: it is trained entirely from scratch, without any pre-training, distillation, or curriculum learning. Our study largely closes the gap between one-step diffusion/flow models and their multi-step predecessors, and we hope it will inspire future work to reconsider the foundations of these powerful models.

我们的 MeanFlow 模型在一步生成建模中表现出强大的经验性能。在 ImageNet 256 $\times$256 [7] 上，我们的方法使用 1-NFE(函数评估次数) 生成，达到了 3.43 的 FID 值。该结果相较于同类先前的最先进方法，显著提升了 50% 到 70% 的相对幅度 (图 1)。此外，我们的方法作为一个独立的生成模型: 完全从零开始训练，无需任何预训练、蒸馏或课程学习。我们的研究在很大程度上缩小了一步扩散/流模型与其多步前身之间的差距，我们希望这能激励未来的工作重新审视这些强大模型的基础。

2 Related Work

2 相关工作

*Work partly done when visiting MIT.

* 部分工作在访问麻省理工学院期间完成。

Diffusion and Flow Matching. Over the past decade, diffusion models [42, 44, 19, 45] have been developed into a highly successful framework for generative modeling. These models progressively add noise to clean data and train a neural network to reverse this process. This procedure involves solving stochastic differential equations (SDE), which is then reformulated as probability flow ordinary differential equations (ODE) [45, 22]. Flow Matching methods [28, 2, 30] extend this framework by modeling the velocity fields that define flow paths between distributions. Flow Matching can also be viewed as a form of continuous-time Normalizing Flows [36].

扩散与流匹配。在过去十年中，扩散模型 [42, 44, 19, 45] 已发展成为生成建模中极为成功的框架。这些模型逐步向干净数据添加噪声，并训练神经网络逆转该过程。该过程涉及求解随机微分方程 (SDE)，随后被重新表述为概率流常微分方程 (ODE)[45, 22]。流匹配方法 [28, 2, 30] 通过建模定义分布间流路径的速度场扩展了该框架。流匹配也可视为连续时间归一化流 (Normalizing Flows)[36] 的一种形式。

Few-step Diffusion/Flow Models. Reducing sampling steps has become an important consideration from both practical and theoretical perspectives. One approach is to distill a pre-trained many-step diffusion model into a few-step model, e.g., [39, 14, 41] or score distillation [32, 50, 53]. Early explorations into training few-step models [46] are built upon the evolution of distillation-based methods. Meanwhile, Consistency Models [46] are developed as a standalone generative model that does not require distillation. These models impose consistency constraints on network outputs at different time steps, encouraging them to produce the same endpoints along the trajectory. Various consistency models and training strategies [46, 43, 15, 31, 49] have been investigated.

少步扩散/流模型。从实际和理论角度来看，减少采样步数已成为重要考量。一种方法是将预训练的多步扩散模型蒸馏为少步模型，例如 [39, 14, 41] 或分数蒸馏 [32, 50, 53]。早期对少步模型训练的探索 [46] 基于蒸馏方法的发展。同时，一致性模型 [46] 作为独立生成模型被提出，无需蒸馏。这些模型对不同时间步的网络输出施加一致性约束，鼓励它们沿轨迹产生相同的终点。各种一致性模型和训练策略 [46, 43, 15, 31, 49] 已被研究。

In recent work, several methods have focused on characterizing diffusion-/flow-based quantities with respect to two time-dependent variables. In [3], a Flow Map is defined as the integral of the flow between two time steps, with several forms of matching losses developed for learning. In comparison to the average velocity our method is based on, the Flow Map corresponds to displacement. Shortcut Models [13] introduce a self-consistency loss function in addition to Flow Matching, which captures relationships between the flows at different discrete time intervals. Inductive Moment Matching [52] models the self-consistency of stochastic interpolants at different time steps.

近期工作中，若干方法聚焦于用两个时间相关变量刻画基于扩散/流的量。在 [3] 中，流映射被定义为两个时间步之间流的积分，并为学习开发了多种匹配损失。相比于我们方法基于的平均速度，流映射对应位移。Shortcut 模型 [13] 在流匹配之外引入了自一致性损失函数，捕捉不同离散时间间隔流之间的关系。归纳矩匹配 [52] 则建模不同时间步随机插值的自一致性。

3 Background: Flow Matching

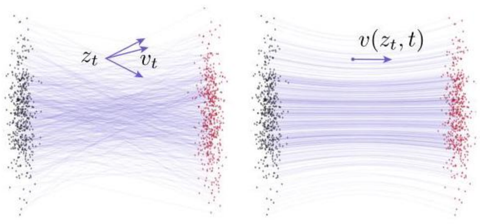
3 背景: 流匹配

Flow Matching $[28, 30, 1]$ is a family of generative models that learn to match the flows, represented by velocity fields, between two probabilistic distributions. Formally, given data $x \sim p_{\text{data}}(x)$ and prior $\epsilon \sim p_{\text{prior}}(\epsilon)$, a flow path can be constructed as $z_t = a_t x + b_t \epsilon$ with time t , where a_t and b_t are predefined schedules. The velocity v_t is defined as $v_t = z_t' = a_t' x + b_t' \epsilon$, where $'$ denotes the time derivative. This velocity is referred to as the conditional velocity in [28], denoted by $v_t = v_t(z_t \mid x)$. See Fig. 2 left. A commonly used schedule is $a_t = 1-t$ and $b_t = t$, which leads to $v_t = \epsilon - x$.

> 流匹配 $[28, 30, 1]$ 是一类生成模型，学习匹配由速度场表示的两个概率分布之间的流。形式上，给定数据 $x \sim p_{\text{data}}(x)$ 和先验 $\epsilon \sim p_{\text{prior}}(\epsilon)$ ，可构造流路径为 $z_t = a_t x + b_t \epsilon$ ，时间为 t ，其中 a_t 和 b_t 为预定义调度。速度 v_t 定义为 $v_t = z_t' = a_t' x + b_t' \epsilon$ ，其中 $'$ 表示时间导数。该速度在 [28] 中称为条件速度，记为 $v_t = v_t(z_t \mid x)$ 。见图 2 左。常用调度为 $a_t = 1-t$ 和 $b_t = t$ ，导致 $v_t = \epsilon - x$ 。

Figure 2: Velocity fields in Flow Matching [28]. Left: con-

> 图 2: 流匹配中的速度场 [28]。左:



ditional flows [28]. A given z_t can arise from different (x, ϵ) pairs, resulting in different conditional velocities v_t . Right: marginal flows [28], obtained by marginalizing over all possible conditional velocities. The marginal velocity field serves as the underlying ground-truth field for network training. All velocities shown here are essentially instantaneous velocities. Illustration follows [12]. (Gray dots: samples from prior; red dots: samples from data.)

> 传统流 [28]。给定的 z_t 可以由不同的 (x, ϵ) 对产生，导致不同的条件速度 v_t 。右图: 边缘流 [28]，通过对所有可能的条件速度进行边缘化得到。边缘速度场作为网络训练的真实基础场。此处显示的所有速度本质上都是瞬时速度。示意图参考 [12]。(灰点: 先验样本；红点: 数据样本。)

Because a given z_t and its v_t can arise from different x and ϵ , Flow Matching essentially models the expectation over all possibilities, called the marginal velocity [28] (Fig. 2 right):

> 由于给定的 z_t 及其 v_t 可以由不同的 x 和 ϵ 产生，流匹配 (Flow Matching) 本质上对所有可能性求期望，称为边缘速度 [28](图 2 右):

$$v(z_t, t) \triangleq \mathbb{E}_{p_t(v_t \mid z_t)}[v_t] . \tag{1}$$

A neural network v_θ parameterized by θ is learned to fit the marginal velocity field: $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t, p_t(z_t)} \left\| v_\theta(z_t, t) - v(z_t, t) \right\|^2$. Although computing this loss function is infeasible due to the marginalization in Eq. (1), it is proposed to instead evaluate the conditional Flow Matching loss [28]: $\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t, x, \epsilon} \left\| v_\theta(z_t, t) - v_t(z_t \mid x) \right\|^2$, where the target v_t is the conditional velocity. Minimizing $\mathcal{L}_{\text{CFM}}$ is equivalent to minimizing $\mathcal{L}_{\text{FM}}$ [28].

学习一个由 θ 参数化的神经网络 v_θ 以拟合边缘速度场: $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p_t(z_t)} \left\| v_\theta(z_t, t) - v(z_t, t) \right\|^2$ 。尽管由于公式 (1) 中的边缘化, 计算该损失函数不可行, 提出改为评估条件流匹配损失 [28]: $\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x,\epsilon} \left\| v_\theta(z_t, t) - v_t(z_t \mid x) \right\|^2$ ，其中目标 v_t 是条件速度。最小化 $\mathcal{L}_{\text{CFM}}$ 等价于最小化 $\mathcal{L}_{\text{FM}}$ [28]。

Given a marginal velocity field $v(z_t, t)$, samples are generated by solving an ODE for z_t :

给定边缘速度场 $v(z_t, t)$ ，通过求解 z_t 的常微分方程 (ODE) 生成样本:

$$\frac{d}{dt} z_t = v(z_t, t) \tag{2}$$

starting from $z_1 = \epsilon \sim p_{\text{prior}}$. The solution can be written as: $z_r = z_t - \int_r^t v(z_\tau, \tau)\, d\tau$, where we use r to denote another time step. In practice, this integral is approximated numerically over discrete time steps. For example, the Euler method, a first-order ODE solver, computes each step as: $z_{t_{i+1}} = z_{t_i} + (t_{i+1} - t_i)\, v(z_{t_i}, t_i)$. Higher-order solvers can also be applied.

从 $z_1 = \epsilon \sim p_{\text{prior}}$ 开始。解可写为: $z_r = z_t - \int_r^t v(z_\tau, \tau)\, d\tau$ ，其中用 r 表示另一个时间步。实际中，该积分通过离散时间步数值近似。例如，一阶 ODE 求解器欧拉法计算每步为: $z_{t_{i+1}} = z_{t_i} + (t_{i+1} - t_i)\, v(z_{t_i}, t_i)$ 。也可应用高阶求解器。

It is worth noting that even when the conditional flows are designed to be straight ("rectified") [28, 30], the marginal velocity field (Eq. (1)) typically induces a curved trajectory. See Fig. 2 for illustration. We also emphasize that this non-straightness is not only a result of neural network approximation, but rather arises from the underlying ground-truth marginal velocity field. When applying coarse discretizations over curved trajectories, numerical ODE solvers lead to inaccurate results.

值得注意的是，即使条件流被设计为直线 (“整流”)[28, 30]，边缘速度场 (公式 (1)) 通常会引起曲线路径。见图 2 示意。我们还强调，这种非直线性不仅是神经网络近似的结果，而是源自真实的边缘速度场。在曲线路径上应用粗糙离散化时，数值 ODE 求解器会导致结果不准确。

4 MeanFlow Models

4 平均流模型

4.1 Mean Flows

4.1 平均流

The core idea of our approach is to introduce a new field representing average velocity, whereas the velocity modeled in Flow Matching represents the instantaneous velocity.

我们方法的核心思想是引入一个表示平均速度的新场，而流匹配中建模的速度表示瞬时速度。

Average Velocity. We define average velocity as the displacement between two time steps t and r (obtained by integration) divided by the time interval. Formally, the average velocity u is:

平均速度。我们定义平均速度为两个时间步 t 和 r 之间的位移 (通过积分获得) 除以时间间隔。形式上，平均速度 u 为:

$$u\left(z_t, r, t\right) \triangleq \frac{1}{t-r} \int_r^t v\left(z_\tau, \tau\right) d\tau. \tag{3}$$

To emphasize the conceptual difference, throughout this paper, we use the notation u to denote average velocity, and v to denote instantaneous velocity. $u\left(z_t, r, t\right)$ is a field that is jointly dependent on(r, t). The field of u is illustrated in Fig. 3. Note that in general, the average velocity u is the result of a functional of the instantaneous velocity v : that is, $u = \mathcal{F}[v] \triangleq \frac{1}{t-r}\int_r^t v d\tau$. It is a field induced by v , not depending on any neural network. Conceptually, just as the instantaneous velocity v serves as the ground-truth field in Flow Matching, the average velocity u in our formulation provides an underlying ground-truth field for learning.

为了强调概念上的区别，本文中我们使用符号 u 表示平均速度，v 表示瞬时速度。$u\left(z_t, r, t\right)$ 是一个同时依赖于 (r, t) 的场。u 场如图 3 所示。注意，一般来说，平均速度 u 是瞬时速度 v 的泛函结果: 即 $u = \mathcal{F}[v] \triangleq \frac{1}{t-r}\int_r^t v d\tau$ 。它是由 v 诱导的场，不依赖于任何神经网络。从概念上讲，正如瞬时速度 v 在流匹配 (Flow Matching) 中作为真实场，本文中的平均速度 u 则为学习提供了一个潜在的真实场。

By definition, the field of u satisfies certain boundary conditions and ”consistency” constraints (generalizing the terminology of [46]). As $r \to t$, we have: $\lim_{r\to t} u = v$. Moreover, a form of ”consistency” is naturally satisfied: taking one larger step over $[r, t]$ is ”consistent” with taking two smaller consecutive steps over $[r, s]$ and $[s, t]$, for any intermediate time s . To see this, observe that $(t-r)\, u\left(z_t, r, t\right) = (s-r)\, u\left(z_s, r, s\right) + (t-s)\, u\left(z_t, s, t\right)$, which follows directly from the additivity of the integral: $\int_r^t v d\tau = \int_r^s v d\tau + \int_s^t v d\tau$. Thus, a network that accurately approximates the true u is expected to satisfy the consistency relation inherently, without the need for explicit constraints.

根据定义，u 场满足某些边界条件和“连贯性”约束 (推广自文献 [46] 的术语)。作为 $r \to t$ ，我们有: $\lim_{r\to t} u = v$ 。此外，一种“连贯性”形式自然成立: 对任意中间时间 s ，在 $[r, t]$ 上迈出一步，与在 $[r, s]$ 和 $[s, t]$ 上连续迈出两步是一致的。为验证此点，观察 $(t-r)\, u\left(z_t, r, t\right) = (s-r)\, u\left(z_s, r, s\right) + (t-s)\, u\left(z_t, s, t\right)$ ，这直接源自积分的可加性: $\int_r^t v d\tau = \int_r^s v d\tau + \int_s^t v d\tau$ 。因此，准确逼近真实 u 的网络预期内在满足连贯性关系，无需显式约束。

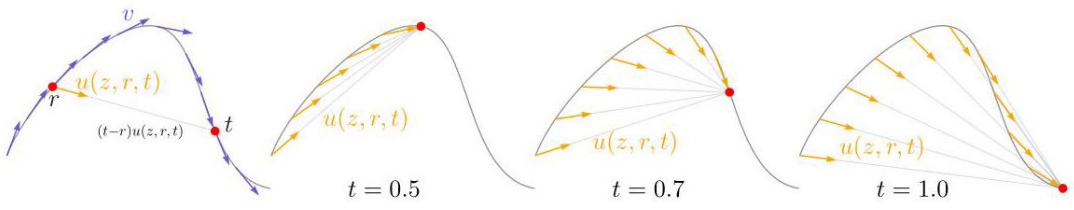


Figure 3: The field of average velocity $u\left(z, r, t\right)$. Leftmost: While the instantaneous velocity v determines the tangent direction of the path, the average velocity $u\left(z, r, t\right)$, defined in Eq. (3), is generally not aligned with v . The average velocity is aligned with the displacement,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

图 3: 平均速度场 $u(z,r,t)$ 。最左图: 瞬时速度 v 决定路径的切线方向，而定义于式 (3) 的平均速度 $u(z,r,t)$ 通常与 v 不一致。平均速度与位移方向一致，即 $(t-r)\,u(z,r,t)$ 。右侧三个子图: $u(z,r,t)$ 场同时依赖于 r 和 t ，此处展示了 $t=0.5, 0.7$ 和 1.0 时的情况。

The ultimate aim of our MeanFlow model will be to approximate the average velocity using a neural network $u_\theta(z_t,r,t)$. This has the notable advantage that, assuming we approximate this quantity accurately, we can approximate the entire flow path using a single evaluation of $u_\theta(\epsilon,0,1)$. In other words, and as we will also demonstrate empirically, the approach is much more amenable to single or few-step generation, as it does not need to explicitly approximate a time integral at inference time, which was required when modeling instantaneous velocity. However, directly using the average velocity defined by Eq. (3) as ground truth for training a network is intractable, as it requires evaluating an integral during training. Our key insight is that the definitional equation of average velocity can be manipulated to construct an optimization target that is ultimately amenable to training, even when only the instantaneous velocity is accessible.

我们 MeanFlow 模型的最终目标是用神经网络 $u_\theta(z_t,r,t)$ 逼近平均速度。这一方法的显著优势在于，假设我们能准确逼近该量，就能通过一次 $u_\theta(\epsilon,0,1)$ 的评估近似整个流路径。换言之，正如我们也将通过实验证明的，这种方法更适合单步或少步生成，因为推理时无需显式逼近时间积分，而瞬时速度建模则需要。然而，直接用式 (3) 定义的平均速度作为训练网络的真实值是不可行的，因为训练时需计算积分。我们的关键见解是，平均速度的定义方程可被变形构造出一个最终适合训练的优化目标，即使仅能访问瞬时速度。

The MeanFlow Identity. To have a formulation amenable to training, we rewrite Eq. (3) as:

MeanFlow 恒等式。为了得到适合训练的表达式，我们将式 (3) 重写为:

$$(t-r)\,u(z_t,r,t)=\int_r^t v(z_\tau,\tau)\,d\tau. \tag{4}$$

Now we differentiate both sides with respect to t , treating r as independent of t . This leads to:

现在对两边关于 t 求导，视 r 为与 t 无关。这得到:

$$\frac{d}{dt}(t-r)\,u(z_t,r,t)=\frac{d}{dt}\int_r^t v(z_\tau,\tau)\,d\tau \Rightarrow u(z_t,r,t)+(t-r)\,\frac{d}{dt}u(z_t,r,t)=v(z_t,t), \tag{5}$$

where the manipulation of the left hand side employs the product rule and the right hand side uses the fundamental theorem of calculus [2] . Rearranging terms, we obtain the identity:

左边的变形使用了乘积法则，右边则利用了微积分基本定理 [2] 。整理项后，我们得到恒等式:

$$\underbrace{u(z_t,r,t)}_{\text{average vel.}}=\underbrace{v(z_t,t)}_{\text{instant. vel.}}-\underbrace{(t-r)\,\frac{d}{dt}u(z_t,r,t)}_{\text{time derivative}} \tag{6}$$

We refer to this equation as the ”MeanFlow Identity”, which describes the relation between v and u . It is easy to show that Eq. (6) and Eq. (4) are equivalent (see Appendix B.3).

我们将此方程称为“平均流恒等式”(MeanFlow Identity)，它描述了 v 与 u 之间的关系。很容易证明方程 (6) 与方程 (4) 是等价的 (见附录 B.3)。

The right hand side of Eq. (6) provides a ”target” form for $u\left(z_t, r, t\right)$, which we will leverage to construct a loss function to train a neural network. To serve as a suitable target, we must also further decompose the time derivative term, which we discuss next.

方程 (6) 的右侧提供了 $u\left(z_t, r, t\right)$ 的“目标”形式，我们将利用它构建损失函数以训练神经网络。为了作为合适的目标，我们还必须进一步分解时间导数项，接下来将进行讨论。

Computing Time Derivative. To compute the $\frac{d}{dt}u$ term in Eq. (6), note that $\frac{d}{dt}$ denotes a total derivative, which can be expanded in terms of partial derivatives:

计算时间导数。为了计算方程 (6) 中的 $\frac{d}{dt}u$ 项，注意 $\frac{d}{dt}$ 表示全导数，可以用偏导数展开:

$$\frac{d}{dt}u\left(z_t, r, t\right)=\frac{dz_t}{dt}\partial_z u+\frac{dr}{dt}\partial_r u+\frac{dt}{dt}\partial_t u. \tag{7}$$

With $\frac{dz_t}{dt}=v\left(z_t, t\right)$ (see Eq. (2)), $\frac{dr}{dt}=0$, and $\frac{dt}{dt}=1$, we have another relation between u and v :

结合 $\frac{dz_t}{dt}=v\left(z_t, t\right)$ (见方程 (2))、$\frac{dr}{dt}=0$ 和 $\frac{dt}{dt}=1$ ，我们得到 u 与 v 之间的另一关系:

$$\frac{d}{dt}u\left(z_t, r, t\right)=v\left(z_t, t\right)\partial_z u+\partial_t u, \tag{8}$$

This equation shows that the total derivative is given by the Jacobian-vector product (JVP) between $[\partial_z u, \partial_r u, \partial_t u]$ (the Jacobian matrix of the function u) and the tangent vector $[v, 0, 1]$. In modern libraries, this can be efficiently computed by the jvp interface, such as torch. func. jvp in PyTorch or jax. jvp in JAX, which we discuss later.

该方程表明，全导数由雅可比矩阵-向量积 (Jacobian-vector product，JVP) 给出，其中 $[\partial_z u, \partial_r u, \partial_t u]$ 是函数 u 的雅可比矩阵，$[v, 0, 1]$ 是切向量。在现代库中，这可以通过 jvp 接口高效计算，如 PyTorch 中的 torch.func.jvp 或 JAX 中的 jax.jvp，后文将讨论。

Training with Average Velocity. Up to this point, the formulations are independent of any network parameterization. We now introduce a model to learn u . Formally, we parameterize a network u_θ and encourage it to satisfy the MeanFlow Identity (Eq. (6)). Specifically, we minimize this objective:

基于平均速度的训练。到目前为止，公式与任何网络参数化无关。现在我们引入一个模型来学习 u 。形式上，我们参数化一个网络 u_θ ，并鼓励其满足平均流恒等式 (方程 (6))。具体地，我们最小化以下目标:

[2] If r depends on t , the Leibniz rule [26] gives: $\frac{d}{dt}\int_r^t v\left(z_\tau, \tau\right) d\tau=v\left(z_t, t\right)-v\left(z_r, r\right)\frac{dr}{dt}$.

[2] 如果 r 依赖于 t ，则 Leibniz 法则 [26] 给出: $\frac{d}{dt}\int_r^t v\left(z_\tau, \tau\right) d\tau=v\left(z_t, t\right)-v\left(z_r, r\right)\frac{dr}{dt}$ 。

$$\mathcal{L}(\theta) = \mathbb{E}\left\| u_\theta(z_t, r, t) - \mathrm{sg}(u_{\mathrm{tgt}}) \right\|_2^2, \tag{9}$$

$$\text{where} \quad u_{\mathrm{tgt}} = v(z_t, t) - (t - r)\left(v(z_t, t)\, \partial_z u_\theta + \partial_t u_\theta\right), \tag{10}$$

The term u_{tgt} serves as the effective regression target, which is driven by Eq. (6). This target uses the instantaneous velocity v as the only ground-truth signal; no integral computation is needed. While the target should involve derivatives of u (that is, ∂u), they are replaced by their parameterized counterparts (that is, ∂u_θ). In the loss function, a stop-gradient (sg) operation is applied on the target u_{tgt} , following common practice [46,43,15,31,13]: in our case, it eliminates the need for ”double backpropagation” through the Jacobian-vector product, thereby avoiding higher-order optimization Despite these practices for optimizability, if u_θ were to achieve zero loss, it is easy to show that it would satisfy the MeanFlow Identity (Eq. (6)), and thus satisfy the original definition (Eq. (3)).

项 u_{tgt} 作为有效的回归目标，由方程 (6) 驱动。该目标仅使用瞬时速度 v 作为唯一的真实信号，无需积分计算。虽然目标应涉及 u 的导数 (即 ∂u)，但它们被其参数化对应物 (即 ∂u_θ) 替代。在损失函数中，目标 u_{tgt} 上应用了停止梯度 (stop-gradient，sg) 操作，遵循常见做法 [46,43,15,31,13]: 在我们的案例中，它消除了通过雅可比矩阵-向量积进行“双重反向传播”的需求，从而避免了高阶优化。尽管采取了这些优化措施，如果 u_θ 达到零损失，很容易证明它将满足平均流恒等式 (方程 (6))，从而满足原始定义 (方程 (3))。

The velocity $v(z_t, t)$ in Eq. (10) is the marginal velocity in Flow Matching [28] (see Fig. 2 right). We follow [28] to replace it with the conditional velocity (Fig. 2 left). With this, the target is:

方程 (10) 中的速度 $v(z_t, t)$ 是流匹配 (Flow Matching)[28] 中的边际速度 (见图 2 右)。我们遵循 [28] 将其替换为条件速度 (图 2 左)。由此，目标为:

$$u_{\mathrm{tgt}} = v_t - (t - r)\left(v_t \partial_z u_\theta + \partial_t u_\theta\right). \tag{11}$$

Recall that $v_t = a_t' x + b_t' \epsilon$ is the conditional velocity [28], and by default, $v_t = \epsilon - x$.

回想 $v_t = a_t' x + b_t' \epsilon$ 是条件速度 [28]，默认情况下，$v_t = \epsilon - x$ 。

Pseudocode for minimizing the loss function Eq. (9) is presented in Alg. 1. Overall, our method is conceptually simple: it behaves similarly to Flow Matching, with the key difference that the matching target is modified by $-(t - r)(v_t \partial_z u_\theta + \partial_t u_\theta)$, arising from our consideration of the average velocity. In particular, note that if we were to restrict to the condition $t = r$, then the second term vanishes, and the method would exactly match standard Flow Matching.

最小化损失函数公式 (9) 的伪代码见算法 1。总体而言，我们的方法在概念上很简单: 其行为类似于流匹配 (Flow Matching)，关键区别在于匹配目标被 $-(t - r)(v_t \partial_z u_\theta + \partial_t u_\theta)$ 修改，这是基于我们对平均速度的考虑。特别地，注意如果我们限制条件为 $t = r$ ，则第二项消失，方法将完全等同于标准流匹配。

In Alg. 1, the jvp operation is highly efficient. In essence, computing $\frac{d}{dt}u$ via jvp requires only a single backward pass, similar to standard back-propagation in neural networks. Because $\frac{d}{dt}u$ is part of the target u_{tgt} and thus subject to stopgrad (w.r.t. θ), the backpropagation for neural network optimization (w.r.t. θ) treats $\frac{d}{dt}u$

as a constant, incurring no higher-order gradient computation. Consequently, jvp introduces only a single extra backward pass, and its cost is comparable to that of backpropagation. In our JAX implementation of Alg. 1, the overhead is less than 20% of the total training time (see appendix).

在算法 1 中，jvp 操作效率极高。本质上，通过 jvp 计算 $\frac{d}{dt}u$ 只需一次反向传播，类似于神经网络中的标准反向传播。由于 $\frac{d}{dt}u$ 是目标 u_{tgt} 的一部分，因此对 θ 的梯度停止传播 (stopgrad)，神经网络优化中对 θ 的反向传播将 $\frac{d}{dt}u$ 视为常数，不涉及高阶梯度计算。因此，jvp 仅引入一次额外的反向传播，其开销与反向传播相当。在我们基于 JAX 实现的算法 1 中，额外开销不到总训练时间的 20%(详见附录)。

Algorithm 1 MeanFlow: Training.

算法 1 MeanFlow: 训练。

Note: in PyTorch and JAX, jvp returns the function output and JVP.

注意: 在 PyTorch 和 JAX 中，jvp 返回函数输出和雅可比向量积 (JVP)。

```
#fn(z, r, t): function to predict u
```

```
#fn(z, r, t): 预测 u 的函数
```

```
#x: training batch
```

```
#x: 训练批次
```

```
t, r = sample_t_r( )
```

```
t, r = sample_t_r( )
```

```
e = randn_like(x)
```

```
e = randn_like(x)
```

$z = (1 - t) * x + t * e$

```
v = e − x
u, dudt = jvp(fn, (z, r, t), (v, 0, 1))
```

```
u, dudt = jvp(fn, (z, r, t), (v, 0, 1))
```

```
u_tgt = v - (t - r) * dudt
```

```
u_tgt = v - (t - r) * dudt
```

```
error = u - stopgrad(u_tgt)
```

```
    error = u - stopgrad(u_tgt)
```

loss = metric(error)

loss = metric(error)

Sampling. Sampling using a MeanFlow model is performed simply by replacing the time integral with the average velocity:

采样。使用 MeanFlow 模型进行采样时，只需用平均速度替代时间积分:

$$z_r = z_t - (t - r)\, u\,(z_t, r, t) \tag{12}$$

In the case of 1-step sampling, we simply have $z_0 = z_1 - u\,(z_1, 0, 1)$, where $z_1 = \epsilon \sim p_{\text{prior}}\,(\epsilon)$. Alg. 2 provides the pseudocode. Although one-step sampling is the main focus on this work, we emphasize that few step sampling is also straightforward given this equation.

在一步采样的情况下，我们简单地有 $z_0 = z_1 - u\,(z_1, 0, 1)$ ，其中 $z_1 = \epsilon \sim p_{\text{prior}}\,(\epsilon)$ 。算法 2 给出了伪代码。虽然本文主要关注一步采样，但我们强调，基于此方程，多步采样同样简单。

Algorithm 2 MeanFlow: 1-step Sampling

算法 2 MeanFlow: 一步采样

```
e = randn ( x_shape )
```

e = randn (x_shape)

$\mathrm{x} = \mathrm{e} - \mathrm{fn}\,(\mathrm{e}, \mathrm{r} = 0, \mathrm{t} = 1)$

Relation to Prior Work. While related to previous one-step generative models [46, 43, 15, 31, 49, 23, 13, 52] , our method provides a more principled framework. At the core of our method is the functional relationship between two underlying fields v and u , which naturally leads to the MeanFlow Identity that u must satisfy (Eq. (6)). This identity does not depend on the introduction of neural networks. In contrast, prior works typically rely on extra consistency constraints, imposed on the behavior of the neural network. Consistency Models [46, 43, 15, 31] are focused on paths anchored at the data side: in our notations, this corresponds to fixing $r \equiv 0$ for any t . As a result, Consistency Models are conditioned on a single time variable, unlike ours. On the other hand, the Shortcut [13] and IMM [52] models are conditioned on two time variables: they introduce additional two-time self-consistency constraints. In contrast, our method is solely driven by the definition of average velocity, and the MeanFlow Identity (Eq. (6)) used for training is naturally derived from this definition, with no extra assumption.

与先前工作的关系。虽然与之前的一步生成模型 [46, 43, 15, 31, 49, 23, 13, 52] 相关，但我们的方法提供了一个更有原则的框架。我们方法的核心是两个潜在场 v 和 u 之间的函数关系，这自然导出了 u 必须满足的 MeanFlow 恒等式 (方程 (6))。该恒等式不依赖于神经网络的引入。相比之下，先前的工作通常依赖于对神经网络行为施加的额外一致性约束。一致性模型 [46, 43, 15, 31] 关注于以数据端为锚点的路径: 用我们的符号表示，即固定任意 t 的 $r \equiv 0$ 。因此，一致性模型仅以单一时间变量为条件，而我们的模型则不同。另一方面，Shortcut[13] 和 IMM[52] 模型以两个时间变量为条件: 它们引入了额外的双时间自一致性约束。相比之下，我们的方法完全由平均速度的定义驱动，训练中使用的 MeanFlow 恒等式 (方程 (6)) 自然从该定义推导而来，无需额外假设。

4.2 Mean Flows with Guidance

4.2 带引导的 Mean Flows

Our method naturally supports classifier-free guidance (CFG) [18]. Rather than naïvely applying CFG at sampling time, which would double NFE, we treat CFG as a property of the underlying ground-truth fields. This formulation allows us to enjoy the benefits of CFG while maintaining the 1-NFE behavior during sampling.

我们的方法天然支持无分类器引导 (classifier-free guidance，CFG)[18]。我们不在采样时简单地应用 CFG(这会使 NFE 翻倍)，而是将 CFG 视为潜在真实场的属性。该表述使我们在采样时保持 1-NFE 的同时，享受 CFG 带来的优势。

Ground-truth Fields. We construct a new ground-truth field v^{cfg} :

真实场。我们构造了一个新的真实场 v^{cfg} :

$$v^{\text{cfg}}(z_t, t \mid \mathbf{c}) \triangleq \omega v(z_t, t \mid \mathbf{c}) + (1-\omega) v(z_t, t), \tag{13}$$

which is a linear combination of a class-conditional and a class-unconditional field:

它是类别条件场和无类别条件场的线性组合:

$$v(z_t, t \mid \mathbf{c}) \triangleq \mathbb{E}_{p_t(v_t \mid z_t, \mathbf{c})}[v_t] \quad \text{and} \quad v(z_t, t) \triangleq \mathbb{E}_{\mathbf{c}}[v(z_t, t \mid \mathbf{c})], \tag{14}$$

where v_t is the conditional velocity [28] (more precisely, sample-conditional velocity in this context). Following the spirit of MeanFlow, we introduce the average velocity u^{cfg} corresponding to v^{cfg} . As per the MeanFlow Identity (Eq. (6)), u^{cfg} satisfies:

其中 v_t 是条件速度 [28](更准确地说，在此上下文中是样本条件速度)。遵循 MeanFlow 的精神，我们引入了对应于 v^{cfg} 的平均速度 u^{cfg} 。根据 MeanFlow 恒等式 (方程 (6))，u^{cfg} 满足:

$$u^{\text{cfg}}(z_t, r, t \mid \mathbf{c}) = v^{\text{cfg}}(z_t, t \mid \mathbf{c}) - (t-r)\frac{d}{dt}u^{\text{cfg}}(z_t, r, t \mid \mathbf{c}). \tag{15}$$

Again, v^{cfg} and u^{cfg} are underlying ground-truth fields that do not depend on neural networks. Here, v^{cfg} , as defined in Eq. (13), can be rewritten as:

同样，v^{cfg} 和 u^{cfg} 是潜在的真实场，不依赖于神经网络。这里，按照方程 (13) 定义的 v^{cfg} 可以重写为:

$$v^{\text{cfg}}\left(z_t, t \mid \mathbf{c}\right) = \omega v\left(z_t, t \mid \mathbf{c}\right) + (1-\omega)\, u^{\text{cfg}}\left(z_t, t, t\right), \tag{16}$$

where we leverage the relation [3] : $v\left(z_t, t\right) = v^{\text{cfg}}\left(z_t, t\right)$, as well as $v^{\text{cfg}}\left(z_t, t\right) = u^{\text{cfg}}\left(z_t, t, t\right)$.

我们利用了关系 [3] : $v\left(z_t, t\right) = v^{\text{cfg}}\left(z_t, t\right)$ 以及 $v^{\text{cfg}}\left(z_t, t\right) = u^{\text{cfg}}\left(z_t, t, t\right)$ 。

Training with Guidance. With Eq. (15) and Eq. (16), we construct a network and its learning target. We directly parameterize u^{cfg} by a function u_θ^{cfg} . Based on Eq. (15), we obtain the objective:

带引导的训练。利用方程 (15) 和方程 (16)，我们构建了网络及其学习目标。我们通过函数 u_θ^{cfg} 直接参数化 u^{cfg} 。基于方程 (15)，我们得到目标函数:

$$\mathcal{L}(\theta) = \mathbb{E}\left\| u_\theta^{\text{cfg}}\left(z_t, r, t \mid \mathbf{c}\right) - \operatorname{sg}\left(u_{\text{tgt}}\right) \right\|_2^2, \tag{17}$$

$$\text{where } u_{\text{tgt}} = \widetilde{v}_t - (t-r)\left(\widetilde{v}_t \partial_z u_\theta^{\text{cfg}} + \partial_t u_\theta^{\text{cfg}}\right) . \tag{18}$$

This formulation is similar to Eq. (9), with the only difference that it has a modified $\widetilde{v}_t$:

该表述与方程 (9) 类似，唯一的区别是它有一个修改后的 $\widetilde{v}_t$:

$$\widetilde{v}_t \triangleq \omega v_t + (1-\omega)\, u_\theta^{\text{cfg}}\left(z_t, t, t\right), \tag{19}$$

which is driven by Eq. (16): the term $v\left(z_t, t \mid \mathbf{c}\right)$ in Eq. (16), which is the marginal velocity, is replaced by the (sample-)conditional velocity v_t , following [28]. If $\omega = 1$, this loss function degenerates to the no-CFG case in Eq. (9).

该 $v\left(z_t, t \mid \mathbf{c}\right)$ 由方程 (16) 驱动: 方程 (16) 中的 $v\left(z_t, t \mid \mathbf{c}\right)$ 项，即边缘速度，被 (样本) 条件速度 v_t 替代，遵循 [28]。如果 $\omega = 1$ ，该损失函数退化为无 CFG 情况下的方程 (9)。

To expose the network u_θ^{cfg} in Eq. (17) to class-unconditional inputs, we drop the class condition with 10% probability, following [18]. Driven by a similar motivation, we can also expose $u_\theta^{\text{cfg}}\left(z_t, t, t\right)$ in Eq. (19) to both class-unconditional and class-conditional versions: the details are in Appendix B.1.

为了使网络中的 u_θ^{cfg} (方程 (17)) 暴露于无类别条件输入，我们以 10% 的概率丢弃类别条件，遵循 [18]。出于类似动机，我们也可以使方程 (19) 中的 $u_\theta^{\text{cfg}}\left(z_t, t, t\right)$ 同时暴露于无类别条件和类别条件版本: 详情见附录 B.1。

Single-NFE Sampling with CFG. In our formulation, u_θ^{cfg} directly models u^{cfg} , which is the average velocity induced by the CFG velocity v^{cfg} (Eq. (13)). As a result, no linear combination is required during sampling: we directly use u_θ^{cfg} for one-step sampling (see Alg. 2), with only a single NFE. This formulation preserves the desirable single-NFE behavior.

使用 CFG 的单次 NFE 采样。在我们的公式中，u_θ^{cfg} 直接建模了由 CFG 速度 v^{cfg} (方程 (13)) 引起的平均速度 u^{cfg} 。因此，采样过程中不需要线性组合: 我们直接使用 u_θ^{cfg} 进行一步采样 (见算法 2)，仅需一次 NFE。该公式保持了理想的单次 NFE 特性。

4.3 Design Decisions

4.3 设计决策

Loss Metrics. In Eq. (9), the metric considered is the squared L2 loss. Following [46, 43, 15], we investigate different loss metrics. In general, we consider the loss function in the form of $\mathcal{L} =\| \Delta\|_2^{2\gamma}$, where Δ denotes the regression error. It can be proven (see [15]) that minimizing $\| \Delta\|_2^{2\gamma}$ is equivalent to minimizing the squared L2 loss $\| \Delta\|_2^2$ with "adapted loss weights". Details are in the appendix. In practice, we set the weight as $w = 1/\left(\| \Delta\|_2^2 + c\right)^p$, where $p = 1-\gamma$ and $c > 0$ (e.g., 10^{-3}). The adaptively weighted loss is $\text{sg}(w) \cdot \mathcal{L}$, with $\mathcal{L} =\| \Delta\|_2^2$. If $p = 0.5$, this is similar to the Pseudo-Huber loss in [43]. We compare different p values in experiments.

损失度量。在方程 (9) 中，考虑的度量是平方 L2 损失。遵循 [46, 43, 15]，我们研究了不同的损失度量。一般而言，我们考虑形式为 $\mathcal{L} =\| \Delta\|_2^{2\gamma}$ 的损失函数，其中 Δ 表示回归误差。可证明 (见 [15]) 最小化 $\| \Delta\|_2^{2\gamma}$ 等价于以"自适应损失权重"最小化平方 L2 损失 $\| \Delta\|_2^2$ 。详情见附录。实际中，我们将权重设为 $w = 1/\left(\| \Delta\|_2^2 + c\right)^p$ ，其中 $p = 1-\gamma$ 和 $c > 0$ (例如，10^{-3})。自适应加权损失为 $\text{sg}(w) \cdot \mathcal{L}$ ，其中 $\mathcal{L} =\| \Delta\|_2^2$ 。若 $p = 0.5$ ，则类似于 [43] 中的伪 Huber 损失。我们在实验中比较了不同的 p 值。

Sampling Time Steps(r, t). We sample the two time steps(r, t)from a predefined distribution We investigate two types of distributions: (i) a uniform distribution,$\mathcal{U}(0,1)$, and (ii) a logit-normal (lognorm) distribution [11], where a sample is first drawn from a normal distribution $\mathcal{N}(\mu,\sigma)$ and then mapped to(0,1)using the logistic function. Given a sampled pair, we assign the larger value to t and the smaller to r . We set a certain portion of random samples with $r = t$.

采样时间步 (r, t)。我们从预定义分布中采样两个时间步 (r, t)。研究了两种分布类型:(i) 均匀分布，$\mathcal{U}(0,1)$ ，以及 (ii) 对数正态 (lognorm) 分布 [11]，先从正态分布 $\mathcal{N}(\mu,\sigma)$ 中采样，再用逻辑函数映射到 (0,1)。给定采样对，我们将较大值赋给 t ，较小值赋给 r 。我们设置一定比例的随机样本满足 $r = t$ 。

Conditioning on(r, t). We use positional embedding [48] to encode the time variables, which are then combined and provided as the conditioning of the neural network. We note that although the field is parameterized by $u_\theta(z_t, r, t)$, it is not necessary for the network to directly condition on(r, t) For example, we can let the network directly condition on $(t, \Delta t)$, with $\Delta t = t - r$. In this case, we have $u_\theta(\cdot, r, t) \triangleq \text{net}(\cdot, t, t-r)$ where net

[3] Observe that: $v^{\text{cfg}}(z_t, t) \triangleq \mathbb{E}_{\mathbf{c}}\left[v^{\text{cfg}}(z_t, t \mid \mathbf{c})\right] = \omega \mathbb{E}_{\mathbf{c}}\left[v(z_t, t \mid \mathbf{c})\right] + (1-\omega) v(z_t, t) = v(z_t, t)$.

[3] 注意: $v^{\text{cfg}}(z_t, t) \triangleq \mathbb{E}_{\mathbf{c}}\left[v^{\text{cfg}}(z_t, t \mid \mathbf{c})\right] = \omega \mathbb{E}_{\mathbf{c}}\left[v(z_t, t \mid \mathbf{c})\right] + (1-\omega) v(z_t, t) = v(z_t, t)$ 。

is the network. The JVP computation is always w.r.t. the function $u_\theta(\cdot, r, t)$. We compare different forms of conditioning in experiments.

基于 (r, t) 的条件。我们使用位置编码 [48] 对时间变量进行编码，然后组合并作为神经网络的条件输入。注意，尽管场由 $u_\theta(z_t, r, t)$ 参数化，但网络不必直接以 (r, t) 为条件。例如，我们可以让网络直接以 $(t, \Delta t)$ 为条件，且 $\Delta t = t - r$ 。此时，有 $u_\theta(\cdot, r, t) \triangleq \mathrm{net}(\cdot, t, t-r)$ ，其中 net 为网络。JVP 计算始终相对于函数 $u_\theta(\cdot, r, t)$ 。我们在实验中比较了不同的条件形式。

5 Experiments

5 实验

Experiment Setting. We conduct our major experiments on ImageNet [7] generation at 256 ×256 resolution. We evaluate Fréchet Inception Distance (FID) [17] on 50 K generated images. We examine the number of function evaluations (NFE) and study 1-NFE generation by default. Following $[34, 13, 52]$, we implement our models on the latent space of a pre-trained VAE tokenizer [37]. For 256×256 images, the tokenizer produces a latent space of $32 \times 32 \times 4$, which is the input to the model Our models are all trained from scratch. Implementation details are in Appendix A.

实验设置。我们在 256 ×256 分辨率下对 ImageNet [7] 生成进行了主要实验。我们在 50 K 生成的图像上评估 Fréchet Inception Distance (FID) [17]。我们考察了函数评估次数 (NFE)，默认研究 1-NFE 生成。遵循 $[34, 13, 52]$ ，我们在预训练的 VAE 分词器 (tokenizer)[37] 的潜在空间上实现模型。对于 256×256 图像，分词器生成一个维度为 $32 \times 32 \times 4$ 的潜在空间，作为模型输入。我们的模型均从零开始训练。实现细节见附录 A。

In our ablation study, we use the ViT-B/4 architecture (namely, ”Base” size with a patch size of 4) [9] as developed in [34], trained for 80 epochs (400K iterations). As a reference, DiT-B/4 in [34] has 68.4 FID, and SiT-B/4 [33] (in our reproduction) has 58.9 FID, both using 250-NFE sampling.

在消融研究中，我们使用 ViT-B/4 架构 (即“Base”大小，patch 大小为 4)[9]，由 [34] 开发，训练 80 个 epoch(40 万次迭代)。作为参考，[34] 中的 DiT-B/4 的 FID 为 68.4，SiT-B/4 [33](我们复现的结果)FID 为 58.9，均使用 250-NFE 采样。

5.1 Ablation Study

5.1 消融研究

We investigate the model properties in Tab. 1, analyzed next:

我们在表 1 中研究模型属性，具体分析如下:

From Flow Matching to Mean Flows. Our method can be viewed as Flow Matching with a modified target (Alg. 1), and it reduces to standard Flow Matching when r always equals t . Tab. 1a compares the ratio of randomly sampling $r \neq t$. A 0% ratio of $r \neq t$ (reducing to Flow Matching) fails to produce reasonable results for 1-NFE generation. A non-zero ratio of $r \neq t$ enables MeanFlow to take effect, yielding meaningful results under 1-NFE generation. We observe that the model balances between learning the instantaneous velocity $(r = t)$ vs. propagating into $r \neq t$ via the modified target Here, the optimal FID is achieved at a ratio of 25%, and a ratio of 100% also yields a valid result.

从流匹配 (Flow Matching) 到均值流 (Mean Flows)。我们的方法可视为带有修改目标的流匹配 (算法 1)，当 r 始终等于 t 时，退化为标准流匹配。表 1a 比较了随机采样 $r \neq t$ 的比例。0% 比例的 $r \neq t$ (即退化为流匹配) 在 1-NFE 生成时无法产生合理结果。非零比例的 $r \neq t$ 使 MeanFlow 生效，在 1-NFE 生成下产生有意义的结果。我们观察到模型在学习瞬时速度 $(r = t)$ 与通过修改目标传播到 $r \neq t$ 之间取得平衡。此处，最佳 FID 在 25% 比例时达到，100% 比例也能得到有效结果。

JVP Computation. The JVP operation Eq. (8) serves as the core relation that connects all(r, t) coordinates. In Tab. 1b, we conduct a destructive comparison in which incorrect JVP computation is intentionally performed. It shows that meaningful results are achieved only when the JVP computation is correct. Notably, the JVP tangent along $\partial_z u$ is d -dimensional, where d is the data dimension (here, $32 \times 32 \times 4$), and the tangents along $\partial_r u$ and $\partial_t u$ are one-dimensional. Nevertheless, these two time variables determine the field u , and their roles are therefore critical even though they are only one-dimensional.

JVP 计算。JVP 操作 (公式 8) 是连接所有 (r, t) 坐标的核心关系。在表 1b 中，我们进行了破坏性对比，故意执行错误的 JVP 计算。结果显示，只有正确计算 JVP 时才能获得有意义的结果。值得注意的是，沿 $\partial_z u$ 方向的 JVP 切线是 d 维的，其中 d 是数据维度 (此处为 $32 \times 32 \times 4$)，而沿 $\partial_r u$ 和 $\partial_t u$ 方向的切线是一维的。尽管如此，这两个时间变量决定了场 u ，因此它们的作用至关重要，尽管维度仅为一维。

Conditioning on(r, t). As discussed in Sec. 4.3, we can represent $u_\theta(z, r, t)$ by various forms of explicit positional embedding, e.g., $u_\theta(\cdot, r, t) \triangleq \text{net}(\cdot, t, t-r)$. Tab. 1c compares these variants. Tab. 1c shows that all variants of(r, t)embeddings studied yield meaningful 1-NFE results, demonstrating the effectiveness of MeanFlow as a framework. Embedding(t, t - r), that is, time and interval, achieves the best result, while directly embedding(r, t)performs almost as well. Notably, even embedding only the interval $t - r$ yields reasonable results.

基于 (r, t) 的条件。正如第 4.3 节所述，我们可以用多种显式位置嵌入形式表示 $u_\theta(z, r, t)$ ，例如 $u_\theta(\cdot, r, t) \triangleq \text{net}(\cdot, t, t-r)$ 。表 1c 比较了这些变体。表 1c 显示，所有研究的 (r, t) 嵌入变体均能产生有意义的 1-NFE 结果，证明了 MeanFlow 作为框架的有效性。嵌入 (t, t - r)，即时间和区间，取得了最佳结果，而直接嵌入 (r, t) 表现几乎同样出色。值得注意的是，即使仅嵌入区间 $t - r$ 也能得到合理结果。

% of $r \neq t$	FID, 1-NFE
0% (= FM)	328.91
25%	61.06
50%	63.14
100%	67.32

% 的 $r \neq t$	FID，1-NFE
0% (= FM)	328.91
25%	61.06
50%	63.14
100%	67.32

(a) Ratio of sampling $r \neq t$. The 0% entry reduces to the standard Flow Matching baseline.

(a) 采样比例 $r \neq t$ 。0% 条目简化为标准流匹配 (Flow Matching) 基线。

jvp tangent	FID, 1-NFE
(v,0,1)	61.06
(v,0,0)	268.06
(v,1,0)	329.22
(v,1,1)	137.96

颈静脉压切线	FID，1-非氟烷 (1-NFE)
(v,0,1)	61.06
(v,0,0)	268.06
(v,1,0)	329.22
(v,1,1)	137.96

(b) JVP computation. The correct jvp tangent is(v,0,1)for Jacobian $(\partial_z u, \partial_r u, \partial_t u)$.

(b) JVP 计算。对于雅可比矩阵 $(\partial_z u, \partial_r u, \partial_t u)$ ， 正确的 JVP 切线是 (v,0,1)。

pos. embed	FID, 1-NFE
(t, r)	61.75
(t, t - r)	61.06
(t, r, t - r)	63.98
$t - r$ only	63.13

位置嵌入	FID，1-NFE
(t, r)	61.75
(t, t - r)	61.06
(t, r, t - r)	63.98
$t - r$ 仅	63.13

(c) Positional embedding. The network is conditioned on the embeddings applied to the specified variables.

(c) 位置嵌入。网络以应用于指定变量的嵌入作为条件。

t, r sampler	FID, 1-NFE
uniform(0,1)	65.90
lognorm(-0.2,1.0)	63.83
lognorm(-0.2,1.2)	64.72
lognorm(-0.4, 1.0)	61.06
lognorm(-0.4,1.2)	61.79

t, r 采样器	FID，1-NFE
均匀分布 (0,1)	65.90
对数正态分布 (-0.2,1.0)	63.83
对数正态分布 (-0.2,1.2)	64.72
对数正态分布 (-0.4, 1.0)	61.06
对数正态分布 (-0.4,1.2)	61.79

p	FID, 1-NFE
0.0	79.75
0.5	63.98
1.0	61.06
1.5	66.57
2.0	69.19

p	FID，1-非萃取物
0.0	79.75
0.5	63.98
1.0	61.06
1.5	66.57
2.0	69.19

ω	FID, 1-NFE
1.0 (w/o cfg)	61.06
1.5	33.33
2.0	20.15
3.0	15.53
5.0	20.75

ω	FID，1-NFE
1.0(无配置)	61.06
1.5	33.33
2.0	20.15
3.0	15.53
5.0	20.75

(d) Time samplers. t and r are sampled (e) Loss metrics. $p = 0$ is squared L2 (f) CFG scale:. Our method supports from the specific sampler. loss. $p = 0.5$ is Pseudo-Huber loss. 1-NFE CFG sampling.

(d) 时间采样器。t 和 r 被采样 (e) 损失度量。$p = 0$ 是平方 L2 损失 (f) CFG 尺度: 我们的方法支持特定采样器的损失。$p = 0.5$ 是伪 Huber 损失。1-NFE CFG 采样。

Table 1: Ablation study on 1-NFE ImageNet 256 $\times$256 generation. FID-50K is evaluated. Default configurations are marked in gray : B/4 backbone, 80-epoch training from scratch.

表 1:1-NFE ImageNet 256 $\times$256 生成的消融研究。评估了 FID-50K。默认配置标记为灰色:B/4 骨干网络，80 轮从头训练。

Time Samplers. Prior work [11] has shown that the distribution used to sample t influences the generation quality. We study the distribution used to sample(r, t)in Tab. 1d. Note that(r, t)are first sampled independently, followed by a post-processing step that enforces $t > r$ by swapping and then caps the proportion of $r \neq t$ to a specified ratio. Tab. 1d reports that a logit-normal sampler performs the best, consistent with observations on Flow Matching [11].

时间采样器。先前工作 [11] 表明用于采样 t 的分布会影响生成质量。我们在表 1d 中研究了用于采样 (r, t) 的分布。注意 (r, t) 首先独立采样，随后通过交换执行后处理步骤以强制执行 $t > r$ ，并将 $r \neq t$ 的比例限制在指定范围内。表 1d 显示对数正态采样器表现最佳，与 Flow Matching[11] 的观察一致。

Loss Metrics. It has been reported [43] that the choice of loss metrics strongly impacts the performance of few-/one-step generation. We study this aspect in Tab. 1e. Our loss metric is implemented via adaptive loss weighting [15] with power p (Sec. 4.3). Tab. 1e shows that $p = 1$ achieves the best result, whereas $p = 0.5$ (similar to Pseudo-Huber loss [43]) also performs competitively. The standard squared L2 loss (here, $p = 0$) underperforms compared to other settings, but still produces meaningful results, consistent with observations in [43].

损失度量。据报道 [43]，损失度量的选择对少步/一步生成性能影响显著。我们在表 1e 中研究了这一方面。我们的损失度量通过自适应损失加权 [15] 实现，幂次为 p (第 4.3 节)。表 1e 显示 $p = 1$ 取得最佳结果，而 $p = 0.5$ (类似伪 Huber 损失 [43]) 也表现出竞争力。标准平方 L2 损失 (此处为 $p = 0$) 表现不及其他设置，但仍产生有意义结果，与 [43] 的观察一致。

Guidance Scale. Tab. 1f reports the results with CFG. Consistent with observations in multi-step generation [34], CFG substantially improves generation quality in our 1-NFE setting too. We emphasize that our CFG formulation (Sec. 4.2) naturally support 1-NFE sampling.

引导尺度。表 1f 报告了 CFG 下的结果。与多步生成中的观察 [34] 一致，CFG 在我们的 1-NFE 设置中也显著提升了生成质量。我们强调，我们的 CFG 公式 (第 4.2 节) 自然支持 1-NFE 采样。

Scalability. Fig. 4 presents the 1-NFE FID results of MeanFlow across larger model sizes and different training durations. Consistent with the behavior of Transformer-based diffusion/flow models (DiT [34] and SiT [33]), MeanFlow models exhibit promising scalability for 1-NFE generation.

可扩展性。图 4 展示了 MeanFlow 在更大模型规模和不同训练时长下的 1-NFE FID 结果。与基于 Transformer 的扩散/流模型 (DiT[34] 和 SiT[33]) 的表现一致，MeanFlow 模型在 1-NFE 生成方面展现出良好的可扩展性。

5.2 Comparisons with Prior Work

5.2 与先前工作的比较

ImageNet 256 $\times$ 256 Comparisons. In Fig. 1 we compare with previous one-step diffusion/flow models, which are also summarized in Tab. 2 (left). Overall, MeanFlow largely outperforms previous methods in its class: it achieves 3.43 FID, which is an over 50% relative improvement vs. IMM's one-step result of 7.77 [52]; if we compare only 1-NFE (not just one-step) generation, MeanFlow has nearly 70% relative improvement vs . the previous state-of-the-art (10.60, Shortcut [13]). Our method largely closes the gap between one-step and many-step diffusion/flow models.

ImageNet 256 $\times$ 256 比较。图 1 中我们与之前的一步扩散/流模型进行了比较，相关内容也汇总于表 2(左侧)。总体上，MeanFlow 在同类方法中表现优异: 其 FID 为 3.43，相较 IMM 一步结果 7.77[52] 提升超过 50%；若仅比较 1-NFE(不仅仅是一步) 生成，MeanFlow 相较之前的最先进方法 (10.60, Shortcut[13]) 有近 70% 的相对提升 vs 。我们的方法大幅缩小了一步与多步扩散/流模型之间的差距。

In 2-NFE generation, our method achieves an FID of 2.20 (Tab. 2, bottom left). This result is on par with the leading baselines of many-step diffusion/flow models, namely, DiT [34] (FID 2.27) and SiT [33] (FID 2.15), both having an NFE of 250 $\times$ 2 (Tab. 2, right), under the same XL/2 backbone. Our results suggest that few-step diffusion/flow models can rival their many-step predecessors. Orthogonal improvements, such as REPA [51], are applicable, which are left for future work.

在 2-NFE 生成中，我们的方法取得了 2.20 的 FID(表 2，左下)。该结果与多步扩散/流模型的领先基线 DiT[34](FID 2.27) 和 SiT[33](FID 2.15) 相当，二者在相同 XL/2 骨干下的 NFE 均为 250 $\times$ 2 (表 2，右侧)。我们的结果表明，少步扩散/流模型能够媲美多步前辈。正交改进如 REPA[51] 可应用，留待未来工作。

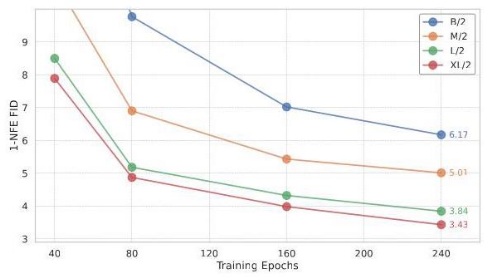


Figure 4: Scalability of MeanFlow models on ImageNet 256 $\times$256.1 -NFE generation FID is reported. All models are trained from scratch. CFG is applied while maintaining the 1-NFE sampling behavior. Our method exhibits promising scalability with respect to model size.

图 4:MeanFlow 模型在 ImageNet 256 $\times$256.1 -NFE 生成上的可扩展性报告了 FID。所有模型均从头训练。应用 CFG 同时保持 1-NFE 采样行为。我们的方法在模型规模方面展现出良好的可扩展性。

method	params	NFE	FID
1-NFE diffusion/flow from scratch			
iCT-XL/2 [43] †	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	3.43
2-NFE diffusion/flow from scratch			
iCT-XL/2 [43] †	675M	2	20.30
iMM-XL/2 [52]	675M	1×2	7.77
MeanFlow-XL/2	676M	2	2.93
MeanFlow-XL/2+	676M	2	2.20

方法	参数	函数评估次数 (NFE)	弗雷歇特距离 (FID)
1-NFE 从零开始的扩散/流			
iCT-XL/2 [43] †	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	3.43
2-NFE 从零开始的扩散/流			
iCT-XL/2 [43] †	675M	2	20.30
iMM-XL/2 [52]	675M	1×2	7.77
MeanFlow-XL/2	676M	2	2.93
MeanFlow-XL/2+	676M	2	2.20

method	params	NFE	FID
$GANs$			
BigGAN [5]	112M	1	6.95
GigaGAN [21]	569M	1	3.45
StyleGAN-XL [40]	166M	1	2.30
autoregressive/masking			
AR w/ VQGAN [10]	227M	1024	26.52
MaskGIT [6]	227M	8	6.18
VAR-d30 [47]	2B	10×2	1.92
MAR-H [27]	943M	256×2	1.55
diffusion/flow			
ADM [8]	554M	250×2	10.94
LDM-4-G [37]	400M	250×2	3.60
SimDiff [20]	2B	512×2	2.77
DiT-XL/2 [34]	675M	250×2	2.27
SiT-XL/2 [33]	675M	250×2	2.06
SiT-XL/2+REPA [51]	675M	250×2	1.42

方法	参数	评估网络前向次数 (NFE)	弗雷歇特推断距离 (FID)
$GANs$			
BigGAN [5]	112M	1	6.95
GigaGAN [21]	569M	1	3.45
StyleGAN-XL [40]	166M	1	2.30
自回归/掩码			
带 VQGAN 的自回归 (AR) [10]	227M	1024	26.52
MaskGIT [6]	227M	8	6.18
VAR-d30 [47]	2B	10×2	1.92
MAR-H [27]	943M	256×2	1.55
扩散/流			
ADM [8]	554M	250×2	10.94
LDM-4-G [37]	400M	250×2	3.60
SimDiff [20]	2B	512×2	2.77
DiT-XL/2 [34]	675M	250×2	2.27
SiT-XL/2 [33]	675M	250×2	2.06
SiT-XL/2+REPA [51]	675M	250×2	1.42

Table 2: Class-conditional generation on ImageNet-256 $\times 256$. All entries are reported with CFG, when applicable. Left: 1-NFE and 2-NFE diffusion/flow models trained from scratch. Right: Other families of generative models as a reference. In both tables, ” $\times 2$ ” indicates that CFG incurs an NFE of 2 per sampling step. Our MeanFlow models are all trained for 240 epochs, except that ”MeanFlow-XL+” is trained for more epochs and with configurations selected for longer training, specified in appendix. † : iCT [43] results are reported by [52].

表 2:ImageNet-256 上的类别条件生成 $\times 256$ 。所有条目均在适用时报告了 CFG。左侧: 从零开始训练的 1-NFE 和 2-NFE 扩散/流模型。右侧: 作为参考的其他生成模型家族。在两张表中,“$\times 2$ ”表示 CFG 在每个采样步骤中产生 2 个 NFE。我们的 MeanFlow 模型均训练了 240 个 epoch,除“MeanFlow-XL+”外,该模型训练周期更长,且配置为适应更长的训练,详见附录。† :iCT [43] 的结果由 [52] 报告。

Notably, our method is self-contained and trained entirely from scratch. It achieves the strong results without using any pre-training, distillation, or the curriculum learning adopted in [43, 15, 31].

值得注意的是,我们的方法是自成体系且完全从零开始训练的。它在未使用任何预训练、蒸馏或 [43, 15, 31] 中采用的课程学习的情况下,取得了优异的结果。

CIFAR-10 Comparisons. We report unconditional generation results on CIFAR-10 [25] (32) in Tab. 3. FID-50K is reported with 1-NFE sampling. All entries are with the same U-net [38] developed from [44] ($\tilde{5}$5M), applied directly on the pixel space. All other competitors are with the EDM-style pre-conditioner [22], and ours has no preconditioner. Implementation details are in the appendix. On this dataset, our method is competitive with prior approaches.

CIFAR-10 比较。我们在表 3 中报告了 CIFAR-10 [25](32) 上的无条件生成结果。FID-50K 采用 1-NFE 采样报告。所有条目均使用相同的 U-net [38]，该网络基于 [44] 开发 (约 5500 万参数)，直接应用于像素空间。其他竞争方法均采用 EDM 风格的预处理器 [22]，而我们的方法无预处理器。实现细节见附录。在该数据集上，我们的方法与先前方法具有竞争力。

method	precond	NFE	FID
iCT [43]	EDM	1	2.83
ECT [15]	EDM	1	3.60
sCT [31]	EDM	1	2.97
IMM [52]	EDM	1	3.20
MeanFlow	none	1	2.92

方法	预条件	非线性方程组求解器 (NFE)	频率识别判别器 (FID)
iCT [43]	经验分布映射 (EDM)	1	2.83
ECT [15]	经验分布映射 (EDM)	1	3.60
sCT [31]	经验分布映射 (EDM)	1	2.97
IMM [52]	经验分布映射 (EDM)	1	3.20
平均流 (MeanFlow)	无	1	2.92

Table 3: Unconditional CIFAR-10.

表 3: 无条件 CIFAR-10。

6 Conclusion

6 结论

We have presented MeanFlow, a principled and effective framework for one-step generation. Broadly speaking, the scenario considered in this work is related to multi-scale simulation problems in physics that may involve a range of scales, lengths, and resolution, in space or time. Carrying out numerical simulation is inherently limited by the ability of computers to resolve the range of scales. Our formulation involves describing the underlying quantity at coarsened levels of granularity, a common theme that underlies many important applications in physics. We hope that our work will bridge research in generative modeling, simulation, and dynamical systems in related fields.

我们提出了 MeanFlow，一种原则性且高效的一步生成框架。广义而言，本工作所考虑的场景与物理中的多尺度模拟问题相关，这些问题可能涉及空间或时间上的多种尺度、长度和分辨率。数值模拟的实施本质上受限于计算机解析尺度范围的能力。我们的公式化方法涉及在粗化的粒度层次上描述底层量，这是物理学中许多重要应用的共同主题。我们希望本工作能架起生成建模、模拟和相关领域动力系统研究之间的桥梁。

Acknowledgement

致谢

We greatly thank Google TPU Research Cloud (TRC) for granting us access to TPUs. Zhengyang Geng is partially supported by funding from the Bosch Center for AI. Zico Kolter gratefully acknowledges Bosch's funding for the lab. Mingyang Deng and Xingjian Bai are partially supported by the MIT-IBM Watson AI Lab funding award. We thank Runqian Wang, Qiao Sun, Zhicheng Jiang, Hanhong Zhao, Yiyang Lu, and Xianbang Wang for their help on the JAX and TPU implementation.

我们非常感谢 Google TPU 研究云 (TRC) 为我们提供 TPU 访问权限。耿正阳部分由博世人工智能中心 (Bosch Center for AI) 资助。Zico Kolter 感谢博世对实验室的资助。邓明阳和白兴建部分由 MIT-IBM Watson AI 实验室资助。我们感谢王润谦、孙乔、姜志成、赵汉宏、陆一阳和王贤邦在 JAX 和 TPU 实现方面的帮助。

References

参考文献

[1] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. arXiv preprint arXiv:2209.15571, 2022. 2

Michael S Albergo 和 Eric Vanden-Eijnden. 利用随机插值构建归一化流。arXiv 预印本 arXiv:2209.15571, 2022. 2

[2] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In International Conference on Learning Representations (ICLR), 2023. 1, 2

Michael Samuel Albergo 和 Eric Vanden-Eijnden. 利用随机插值构建归一化流。发表于国际学习表征会议 (ICLR)，2023. 1, 2

[3] Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching. arXiv preprint arXiv:2406.07507, 2024. 2

Nicholas M Boffi, Michael S Albergo 和 Eric Vanden-Eijnden. 流映射匹配。arXiv 预印本 arXiv:2406.07507, 2024. 2

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.16

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne 和 Qiao Zhang. JAX:Python+NumPy 程序的可组合变换，2018。网址 http://github.com/google/jax.16

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In International Conference on Learning Representations (ICLR), 2019.9

Andrew Brock, Jeff Donahue 和 Karen Simonyan. 大规模 GAN 训练用于高保真自然图像合成。发表于国际学习表征会议 (ICLR)，2019.9

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 9

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu 和 William T Freeman. Maskgit: 掩码生成图像变换器。发表于 IEEE 计算机视觉与模式识别会议 (CVPR)，2022. 9

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009. 2, 7

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li 和 Li Fei-Fei. Imagenet: 大规模分层图像数据库。发表于 IEEE 计算机视觉与模式识别会议 (CVPR)，2009. 2, 7

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. Neural Information Processing Systems (NeurIPS), 34, 2021. 9

Prafulla Dhariwal 和 Alexander Nichol. 扩散模型在图像合成上超越 GAN。神经信息处理系统大会 (NeurIPS)，34, 2021. 9

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations (ICLR), 2021. 7, 14

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit 和 Neil Houlsby. 一张图像胜过 16x16 个词: 大规模图像识别的变换器。发表于国际学习表征会议 (ICLR), 2021. 7, 14

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.9

Patrick Esser, Robin Rombach 和 Bjorn Ommer. 驯服变换器用于高分辨率图像合成。发表于 IEEE 计算机视觉与模式识别会议 (CVPR)，2021.9

[11] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In Forty-first international conference on machine learning, 2024. 1, 7, 8

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, 等. 用于高分辨率图像合成的校正流变换器的扩展. 载于第 41 届国际机器学习大会, 2024. 1, 7, 8

[12] Tor Fjelde, Emile Mathieu, and Vincent Dutordoir. An introduction to flow matching. https: //mlg.eng.cam.ac.uk/blog/2024
matching.html, January 2024. Cambridge Machine Learning Group Blog. 3

Tor Fjelde, Emile Mathieu, 和 Vincent Dutordoir. 流匹配简介. https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html, 2024 年 1 月. 剑桥机器学习小组博客. 3

[13] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In International Conference on Learning Representations (ICLR), 2025. 1, 2, 5, 6, 7, 8, 9

Kevin Frans, Danijar Hafner, Sergey Levine, 和 Pieter Abbeel. 通过捷径模型实现一步扩散. 载于国际学习表征会议 (ICLR), 2025. 1, 2, 5, 6, 7, 8, 9

[14] Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. Neural Information Processing Systems (NeurIPS), 36, 2024. 2

Zhengyang Geng, Ashwini Pokle, 和 J Zico Kolter. 通过深度平衡模型实现一步扩散蒸馏. 神经信息处理系统大会 (NeurIPS), 第 36 届, 2024. 2

[15] Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. arXiv preprint arXiv:2406.14548, 2024. 1, 2, 5, 6, 7, 8, 9, 15

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, 和 J Zico Kolter. 一致性模型简化方法. arXiv 预印本 arXiv:2406.14548, 2024. 1, 2, 5, 6, 7, 8, 9, 15

[16] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017. 14

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, 和 Kaiming He. 精确的大批量随机梯度下降:1 小时内训练 ImageNet. arXiv 预印本 arXiv:1706.02677, 2017. 14

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. Neural Information Processing Systems (NeurIPS), 2017. 7

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, 和 Sepp Hochreiter. 通过双时间尺度更新规则训练的 GAN 收敛到局部纳什均衡. 神经信息处理系统大会 (NeurIPS), 2017. 7

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598, 2022. 2, 6, 14, 15

Jonathan Ho 和 Tim Salimans. 无分类器扩散引导. arXiv 预印本 arXiv:2207.12598, 2022. 2, 6, 14, 15

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Neural Information Processing Systems (NeurIPS), 2020. 1, 2

Jonathan Ho, Ajay Jain, 和 Pieter Abbeel. 去噪扩散概率模型. 神经信息处理系统大会 (NeurIPS), 2020. 1, 2

[20] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In International Conference on Machine Learning (ICML), 2023. 9

Emiel Hoogeboom, Jonathan Heek, 和 Tim Salimans. 简单扩散: 端到端高分辨率图像扩散. 载于国际机器学习大会 (ICML), 2023. 9

[21] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 9

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, 和 Taesung Park. 扩展 GAN 用于文本到图像合成. 载于 IEEE 计算机视觉与模式识别会议 (CVPR), 2023. 9

[22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Neural Information Processing Systems (NeurIPS), 2022. 2, 9, 14

Tero Karras, Miika Aittala, Timo Aila, 和 Samuli Laine. 阐明基于扩散的生成模型设计空间. 载于神经信息处理系统大会 (NeurIPS), 2022. 2, 9, 14

[23] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In International Conference on Learning Representations (ICLR), 2024. 5

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, 和 Stefano Ermon. 一致性轨迹模型: 学习扩散的概率流常微分方程轨迹. 载于国际学习表征会议 (ICLR), 2024. 5

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), 2015. 14

Diederik P. Kingma 和 Jimmy Ba. Adam: 一种随机优化方法. 载于国际学习表征会议 (ICLR), 2015. 14

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https: //www.cs.toronto.edu/k̃riz/cifar.h
9

Alex Krizhevsky. 从小图像中学习多层特征. 2009. 网址 https://www.cs.toronto.edu/k̃riz/cifar.html. 9

[26] Gottfried Wilhelm Leibniz. Epistola LXXI ad johannem bernoullium, 5 aug 1697. In Johann Bernoulli, editor, Virorum celeberrimorum G. G. Leibnitti et Johannis Bernoulli Commercium philosophicum et mathematicum, volume I, pages 368-370. Marc-Michel Bousquet, Lausanne & Geneva, 1745. URL https://archive.org/details/bub_gb_103
explicit statement of the Leibniz integral rule. 4

戈特弗里德·威廉·莱布尼茨 (Gottfried Wilhelm Leibniz)。致约翰·伯努利 (Johann Bernoulli) 的第 71 封信，1697 年 8 月 5 日。载于约翰·伯努利主编，《著名学者 G. G. 莱布尼茨与约翰·伯努利的哲学与数学交流》，第一卷，第 368-370 页。Marc-Michel Bousquet，洛桑与日内瓦，1745 年。网址 https://archive.org/details/bub_gb_103w0MxjoF8C/page/368。莱布尼茨积分法则的首次明确表述。4

[27] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. Neural Information Processing Systems (NeurIPS), 2024.9

李天宏，田永龙，李鹤，邓明阳，何凯明。无向量量化的自回归图像生成。神经信息处理系统大会 (NeurIPS)，2024 年。9

[28] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In International Conference on Learning Representations (ICLR),2023.1,2,3,5,6

亚伦·利普曼 (Yaron Lipman)，瑞奇·T. Q. 陈 (Ricky T. Q. Chen)，赫利·本-哈穆 (Heli Ben-Hamu)，马克西米利安·尼克尔 (Maximilian Nickel)，马修·勒 (Matthew Le)。用于生成建模的流匹配。国际学习表征会议 (ICLR)，2023 年。1,2,3,5,6

[29] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.14

亚伦·利普曼,马顿·哈瓦西,彼得·霍尔德里斯,内塔·绍尔,马特·勒,布赖恩·卡勒,瑞奇·T. Q. 陈,大卫·洛佩兹-帕兹, 赫利·本-哈穆, 伊泰·加特。流匹配指南与代码, 2024 年。网址 https://arxiv.org/abs/2412.06264。14

[30] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In International Conference on Learning Representations (ICLR), 2023. 1, 2, 3

刘星超，龚承岳，刘强。流直且快: 学习生成与传递数据的校正流。国际学习表征会议 (ICLR)，2023 年。1,2,3

[31] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In International Conference on Learning Representations (ICLR), 2025. 1, 2, 5, 6, 9

陆成，宋洋。简化、稳定与扩展连续时间一致性模型。国际学习表征会议 (ICLR)，2025 年。1,2,5,6,9

[32] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. Neural Information Processing Systems (NeurIPS), 2024. 2

罗伟健，胡天阳，张世峰，孙嘉诚，李正国，张志华。Diff-instruct: 一种从预训练扩散模型迁移知识的通用方法。神经信息处理系统大会 (NeurIPS)，2024 年。2

[33] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In European Conference on Computer Vision (ECCV), 2024. 1, 7, 8, 9

马南业，马克·戈德斯坦，迈克尔·S·阿尔贝戈，尼古拉斯·M·博菲，埃里克·范登-艾延登，谢赛宁。SIT: 探索基于流和扩散的生成模型与可扩展插值变换器。欧洲计算机视觉大会 (ECCV)，2024 年。1,7,8,9

[34] William Peebles and Saining Xie. Scalable diffusion models with transformers. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 7, 8, 9, 14

威廉·皮布尔斯，谢赛宁。基于变换器的可扩展扩散模型。IEEE 计算机视觉与模式识别会议 (CVPR)，2023 年。7,8,9,14

[35] Adam Polyak,, et al. Movie Gen: A cast of media foundation models, 2025. 1

亚当·波利亚克，等。Movie Gen: 一组媒体基础模型，2025 年。1

[36] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In International Conference on Machine Learning (ICML), 2015. 2

达尼洛·雷岑德，沙基尔·穆罕默德。基于归一化流的变分推断。国际机器学习会议 (ICML)，2015 年。2

[37] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 7, 9

罗宾·罗姆巴赫，安德烈亚斯·布拉特曼，多米尼克·洛伦茨，帕特里克·埃塞尔，比约恩·奥默。基于潜在扩散模型的高分辨率图像合成。IEEE 计算机视觉与模式识别会议 (CVPR)，2021 年。7,9

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention (MICCAI), 2015. 9, 14

奥拉夫·罗内伯格，菲利普·费舍尔，托马斯·布罗克斯。U-net: 用于生物医学图像分割的卷积网络。医学图像计算与计算机辅助干预会议 (MICCAI)，2015 年。9,14

[39] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In International Conference on Learning Representations (ICLR), 2022. 2

蒂姆·萨利曼斯，乔纳森·霍。扩散模型快速采样的渐进蒸馏。国际学习表征会议 (ICLR)，2022 年。2

[40] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In ACM Transactions on Graphics (SIGGRAPH), 2022. 9

阿克塞尔·绍尔，卡佳·施瓦茨，安德烈亚斯·盖格。StyleGAN-XL: 将 StyleGAN 扩展到大型多样化数据集。ACM 图形学交易 (SIGGRAPH)，2022 年。9

[41] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In European Conference on Computer Vision (ECCV), 2024. 2

阿克塞尔·绍尔，多米尼克·洛伦茨，安德烈亚斯·布拉特曼，罗宾·罗姆巴赫。对抗扩散蒸馏。欧洲计算机视觉大会 (ECCV)，2024 年。2

[42] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International Conference on Machine Learning (ICML), 2015. 1, 2

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, 和 Surya Ganguli. 利用非平衡热力学的深度无监督学习。发表于国际机器学习大会 (ICML)，2015 年。1, 2

[43] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In International Conference on Learning Representations (ICLR), 2024. 1, 2, 5, 6, 7, 8, 9, 15

Yang Song 和 Prafulla Dhariwal. 训练一致性模型的改进技术。发表于国际学习表征会议 (ICLR), 2024 年。1, 2, 5, 6, 7, 8, 9, 15

[44] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. Neural Information Processing Systems (NeurIPS), 2019. 1, 2, 9, 14

Yang Song 和 Stefano Ermon. 通过估计数据分布梯度的生成建模。神经信息处理系统大会 (NeurIPS), 2019 年。1, 2, 9, 14

[45] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In International Conference on Learning Representations (ICLR), 2021. 2

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, 和 Ben Poole. 通过随机微分方程的基于分数的生成建模。发表于国际学习表征会议 (ICLR)，2021 年。2

[46] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In International Conference on Machine Learning (ICML), 2023. 1, 2, 3, 5, 6, 7

Yang Song, Prafulla Dhariwal, Mark Chen, 和 Ilya Sutskever. 一致性模型。发表于国际机器学习大会 (ICML)，2023 年。1, 2, 3, 5, 6, 7

[47] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. Neural Information Processing Systems (NeurIPS), 2024. 9

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, 和 Liwei Wang. 视觉自回归建模: 通过下一级尺度预测实现可扩展图像生成。神经信息处理系统大会 (NeurIPS)，2024 年。9

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Neural Information Processing Systems (NeurIPS), 2017. 7

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, 和 Illia Polosukhin. 注意力机制即一切。神经信息处理系统大会 (NeurIPS)，2017 年。7

[49] Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. arXiv preprint arXiv:2407.02398, 2024. 2, 5

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, 和 Bin Cui. 一致性流匹配: 通过速度一致性定义直流。arXiv 预印本 arXiv:2407.02398, 2024 年。2, 5

[50] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 2

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, 和 Taesung Park. 一步扩散与分布匹配蒸馏。发表于 IEEE 计算机视觉与模式识别会议 (CVPR)，2024 年。2

[51] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In International Conference on Learning Representations (ICLR), 2025. 9

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, 和 Saining Xie. 生成的表示对齐: 训练扩散变换器比你想象的更简单。发表于国际学习表征会议 (ICLR)，2025 年。9

[52] Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. arXiv preprint arXiv:2503.07565, 2025. 1, 2, 5, 6, 7, 8, 9

Linqi Zhou, Stefano Ermon, 和 Jiaming Song. 归纳矩匹配。arXiv 预印本 arXiv:2503.07565，2025 年。1, 2, 5, 6, 7, 8, 9

[53] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In International Conference on Machine Learning (ICML), 2024. 2

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, 和 Hai Huang. 分数身份蒸馏: 预训练扩散模型的指数级快速蒸馏实现一步生成。发表于国际机器学习大会 (ICML)，2024 年。2

Appendix

附录

A Implementation

A 实现

Table 4: Configurations on ImageNet 256 $\times$256 . B/4 is our ablation model.

表 4:ImageNet 256 上的配置 $\times$256 。B/4 是我们的消融模型。

configs	B/4	B/2	M/2	L/2	XL/2	XL/2+
params (M)	131	131	497.8	459	676	676
FLOPs (G)	5.6	23.1	54.0	119.0	119.0	119.0
depth	12	12	16	24	28	28
hidden dim	768	768	1024	1024	1152	1152
heads	12	12	16	16	16	16
patch size	4×4	2×2	2×2	2×2	2×2	2×2
epochs	80	240	240	240	240	1000
batch size	256					
dropout				0.0		
optimizer	Adam [24]					
lr schedule	constant					
lr	0.0001 (0.9, 0.95)					
Adam (β_1, β_2)						
weight decay				0.0		
ema decay				0.9999		
ratio of $r \neq t$	Tab. 1a			25%		
(r, t)cond	Tab. 1c		(t, t - r)			
(r, t)sampler	Tab. 1d		lognorm(-0.4,1.0)			
p for adaptive weight	Tab. 1e		1.0			
CFG effective scale ω'	Tab. 1f	2.0	2.0	2.5	2.5	2.0
CFG ω , Eq. (21)	$\omega = \omega'$	1.0	1.0	0.2	0.2	1.0
CFG κ , Eq. (21)				$\kappa = 1 - \omega/\omega'$		
CFG cls-cond drop [18]				0.1 [18]		
CFG triggered if t is in:	$[0.0, 1.0]$	$[0.0, 1.0]$	$[0.0, 1.0]$	$[0.0, 0.8]$	$[0.0, 0.75]$	$[0.3, 0.8]$

配置	B/4	B/2	M/2	L/2	XL/2	XL/2+
参数量 (百万)	131	131	497.8	459	676	676
浮点运算次数 (十亿次)	5.6	23.1	54.0	119.0	119.0	119.0
深度	12	12	16	24	28	28
隐藏维度	768	768	1024	1024	1152	1152
头数	12	12	16	16	16	16
分块大小	4×4	2×2	2×2	2×2	2×2	2×2
训练轮数	80	240	240	240	240	1000
批量大小	256					
丢弃率				0.0		
优化器	Adam [24]					
学习率调度	恒定					
学习率	0.0001 (0.9, 0.95)					
Adam (β_1, β_2)						
权重衰减				0.0		
指数移动平均衰减				0.9999		
$r \neq t$ 比例	表 1a			25%		
(r, t) 条件	表 1c		(t, t - r)			
(r, t) 采样器	表 1d		对数正态分布 (-0.4,1.0)			
用于自适应权重的 p	表 1e		1.0			
CFG 有效尺度 ω'	表 1f	2.0	2.0	2.5	2.5	2.0
CFG ω， 方程 (21)	$\omega = \omega'$	1.0	1.0	0.2	0.2	1.0
CFG κ， 方程 (21)				$\kappa = 1 - \omega/\omega'$		
CFG 分类条件丢弃 [18]				0.1 [18]		
当 t 位于以下范围时触发 CFG:	$[0.0, 1.0]$	$[0.0, 1.0]$	$[0.0, 1.0]$	$[0.0, 0.8]$	$[0.0, 0.75]$	$[0.3, 0.8]$

ImageNet 256×256 . We use a standard VAE tokenizer to extract the latent representations. [4] The latent size is $32 \times 32 \times 4$, which is the input to the model. The backbone architectures follow DiT [34], which are based on ViT [9] with adaLN-Zero [34] for conditioning. To condition on two time variables $(e.g., (r, t))$, we apply positional embedding on each time variable, followed by a 2-layer MLP, and summed. We keep the the DiT architecture blocks untouched, while architectural improvements are orthogonal and possible. The configuration specifics are in Tab. 4.

ImageNet 256×256 。我们使用标准的 VAE 分词器提取潜在表示。[4] 潜在向量大小为 $32 \times 32 \times 4$ ，这是模型的输入。主干架构遵循 DiT [34]，基于 ViT [9]，并采用 adaLN-Zero [34] 进行条件控制。为了对两个时间变量 $(e.g., (r, t))$ 进行条件控制，我们对每个时间变量应用位置嵌入，随后通过两层多层感知机 (MLP) 并求和。我们保持 DiT 架构模块不变，架构改进是正交且可行的。具体配置见表 4。

CIFAR-10. We experiment with class-unconditional generation on CIFAR-10. Our implementation follows standard Flow Matching practice [29]. The input to the model is $32 \times 32 \times 3$ in the pixel space. The network is a U-net [38] developed from [44] ($\tilde{5}$5M), which is commonly used by other baselines we compare. We apply positional embedding on the two time variables (here,(t, t - r)) and concatenate them for conditioning. We do not use any EDM preconditioner [22].

CIFAR-10。我们在 CIFAR-10 上进行无类别条件生成实验。我们的实现遵循标准的流匹配 (Flow Matching) 方法 [29]。模型输入为像素空间中的 $32 \times 32 \times 3$ 。网络为基于 [44] 开发的 U-net [38](约 5500 万参数)，这是我们比较的其他基线常用的结构。我们对两个时间变量 (此处为 (t, t - r)) 应用位置嵌入并拼接以进行条件控制。我们未使用任何 EDM 预处理器 [22]。

We use Adam with learning rate 0.0006, batch size 1024, $(\beta_1, \beta_2) = (0.9, 0.999)$, dropout 0.2, weight decay 0, and EMA decay of 0.99995 . The model is trained for 800K iterations (with 10K warm-up [16]). The(r, t)sampler is lognorm $(-2.0, 2.0)$. The ratio of sampling $r \neq t$ is 75%. The power p for adaptive weighting is 0.75 . Our data augmentation setup follows [22], with vertical flipping and rotation disabled.

我们使用 Adam 优化器，学习率为 0.0006，批量大小 1024，$(\beta_1, \beta_2) = (0.9, 0.999)$ ，丢弃率 0.2，权重衰减为 0，EMA 衰减为 0.99995。模型训练 800K 次迭代 (含 10K 次预热 [16])。(r, t) 采样器为 lognorm $(-2.0, 2.0)$ 。采样比例 $r \neq t$ 为 75%。自适应加权的幂指数 p 为 0.75。数据增强设置遵循 [22]，禁用垂直翻转和旋转。

κ	FID, 1-NFE
0.0	20.15
0.5	19.15
0.8	19.10
0.9	18.63
0.95	19.17

κ	FID，1-非挥发性存储器
0.0	20.15
0.5	19.15
0.8	19.10
0.9	18.63
0.95	19.17

Table 5: Improved CFG for MeanFlow. κ is as defined in Eq. (20), whose goal is to enable both class-conditional $u^{\mathrm{cfg}}(\cdot \mid \mathbf{c})$ and class-unconditional $u^{\mathrm{cfg}}(\cdot)$ to appear in the target. In this table, we fix the effective guidance scale ω' , given by $\omega' = \omega / (1 - \kappa)$, as 2.0 . Accordingly, for different κ values, we set ω by $\omega = (1 - \kappa) \cdot \omega'$. If $\kappa = 0$, it falls back to the CFG case in Eq. (19) (see also Tab. 1f). Similar to standard CFG's practice of randomly dropping class conditions [18], we observe that mixing class-conditional and class-unconditional u^{cfg} in our target improves generation quality.

[4] https://huggingface.co/pcuenq/sd-vae-ft-mse-flax

[4] https://huggingface.co/pcuenq/sd-vae-ft-mse-flax

表 5:MeanFlow 的改进 CFG。κ 如公式 (20) 所定义，目的是使目标中同时出现类别条件 $u^{\text{cfg}}(\cdot \mid \mathbf{c})$ 和类别无关 $u^{\text{cfg}}(\cdot)$ 。在本表中，我们将有效引导尺度 ω' (由 $\omega' = \omega/(1-\kappa)$ 给出) 固定为 2.0。相应地，对于不同的 κ 值，我们通过 $\omega = (1-\kappa)\cdot\omega'$ 设置 ω 。如果 $\kappa = 0$ ，则退回到公式 (19) 中的 CFG 情况 (参见表 1f)。类似于标准 CFG 中随机丢弃类别条件的做法 [18]，我们观察到在目标中混合类别条件和类别无关的 u^{cfg} 能提升生成质量。

B Additional Technical Details

B 附加技术细节

B.1 Improved CFG for MeanFlow

B.1 MeanFlow 的改进 CFG

In Sec. 4.2, we have discussed how to naturally extend our method to supporting CFG. The only change needed is to revise the target by Eq. (19). We notice that only the class-unconditional u^{cfg} is presented in Eq. (19). In the original CFG [18], it is a standard practice to mix class-conditional and class-unconditional predictions, approached by random dropping. We observe that a similar idea can be applied to our regression target as well.

在第 4.2 节中，我们讨论了如何自然地将方法扩展以支持 CFG。唯一需要的改动是通过公式 (19) 修正目标。我们注意到公式 (19) 中仅出现了类别无关的 u^{cfg} 。在原始 CFG[18] 中，混合类别条件和类别无关预测是标准做法，通常通过随机丢弃实现。我们观察到类似的思路也适用于我们的回归目标。

Formally, we introduce a mixing scale κ and rewrite Eq. (16) as:

形式上，我们引入混合比例 κ ，并将公式 (16) 重写为:

$$v^{\text{cfg}}(z_t, t \mid \mathbf{c}) = \omega v(z_t, t \mid \mathbf{c}) + \kappa u^{\text{cfg}}(z_t, t, t \mid \mathbf{c}) + (1-\omega-\kappa)\, u^{\text{cfg}}(z_t, t, t). \tag{20}$$

Here, the role of κ is to mix with $u^{\text{cfg}}(\cdot \mid \mathbf{c})$ on the right hand side. We can show that Eq. (20) satisfies the original CFG formulation (Eq. (13)) with the effective guidance scale of $\omega' = \frac{\omega}{1-\kappa}$, leveraging the relation $v(z_t, t) = v^{\text{cfg}}(z_t, t)$ and $v^{\text{cfg}}(z_t, t) = u^{\text{cfg}}(z_t, t, t)$ that we have used for deriving Eq. (16). With this, Eq. (19) is rewritten as

这里，κ 的作用是在右侧与 $u^{\text{cfg}}(\cdot \mid \mathbf{c})$ 混合。我们可以证明公式 (20) 满足原始 CFG 公式 (公式 (13))，其有效引导尺度为 $\omega' = \frac{\omega}{1-\kappa}$ ，利用了我们推导公式 (16) 时使用的关系 $v(z_t, t) = v^{\text{cfg}}(z_t, t)$ 和 $v^{\text{cfg}}(z_t, t) = u^{\text{cfg}}(z_t, t, t)$ 。据此，公式 (19) 被重写为

$$\widetilde{v}_t \triangleq \omega\underbrace{(\epsilon - x)}_{\text{sample } v_t} + \underbrace{\kappa u_\theta^{\text{cfg}}(z_t, t, t \mid \mathbf{c})}_{\text{cls-cond output}} + \underbrace{(1-\omega-\kappa)\, u_\theta^{\text{cfg}}(z_t, t, t)}_{\text{cls-uncond output}}. \tag{21}$$

The loss function is the same as defined in Eq. (17).

损失函数与公式 (17) 中定义的相同。

The influence of introducing κ is explored in Tab. 5, where we fix the effective guidance scale ω' as 2.0 and vary κ . It shows that mixing by κ can further improve generation quality. We note that the ablation in Tab. 1f in the main paper did not involve this improvement, i.e., κ was set as 0 .

引入 κ 的影响在表 5 中进行了探讨，我们将有效引导尺度 ω' 固定为 2.0，改变 κ。结果显示，通过 κ 混合可以进一步提升生成质量。我们注意到主文中表 1f 的消融实验未涉及此改进，即 κ 被设为 0。

B.2 Loss Metrics

B.2 损失度量

The squared L2 loss is given by $\mathcal{L} = \| \Delta\|_2^2$, where $\Delta = u_\theta - u_{\text{tgt}}$ denotes the regression error. Generally, one can adopt the powered L2 loss $\mathcal{L}_\gamma = \| \Delta\|_2^{2\gamma}$, where γ is a user-specified hyper-parameter. Minimizing this loss is equivalent to minimizing an adaptively weighted squared L2 loss (see [15]): $\frac{d}{d\theta}\mathcal{L}_\gamma = \gamma\left(\| \Delta\|_2^2\right)^{(\gamma-1)} \cdot \frac{d\|\Delta\|_2^2}{d\theta}$. This can be viewed as weighting the squared L2 loss $\left(\| \Delta\|_2^2\right)$ by a loss-adaptive weight $\lambda \propto \| \Delta\|_2^{2(\gamma-1)}$. In practice, we follow [15] and weight by:

平方 L2 损失定义为 $\mathcal{L} = \| \Delta\|_2^2$，其中 $\Delta = u_\theta - u_{\text{tgt}}$ 表示回归误差。通常，可以采用幂次 L2 损失 $\mathcal{L}_\gamma = \| \Delta\|_2^{2\gamma}$，其中 γ 是用户指定的超参数。最小化该损失等价于最小化自适应加权的平方 L2 损失 (参见 [15]): $\frac{d}{d\theta}\mathcal{L}_\gamma = \gamma\left(\| \Delta\|_2^2\right)^{(\gamma-1)} \cdot \frac{d\|\Delta\|_2^2}{d\theta}$ 。这可以看作是用损失自适应权重 $\lambda \propto \| \Delta\|_2^{2(\gamma-1)}$ 对平方 L2 损失 $\left(\| \Delta\|_2^2\right)$ 进行加权。实际中，我们遵循 [15]，权重为:

$$w = 1/\left(\| \Delta\|_2^2 + c\right)^p, \tag{22}$$

where $p = 1 - \gamma$ and $c > 0$ is a small constant to avoid division by zero. If $p = 0.5$, this is similar to the Pseudo-Huber loss in [43]. The adaptively weighted loss is sg $(w) \cdot \mathcal{L}$, where sg denotes the stop-gradient operator.

其中 $p = 1 - \gamma$ 和 $c > 0$ 是避免除零的小常数。如果 $p = 0.5$，这类似于 [43] 中的伪 Huber 损失 (Pseudo-Huber loss)。自适应加权损失为 sg $(w) \cdot \mathcal{L}$，其中 sg 表示停止梯度操作符。

B.3 On the Sufficiency of the MeanFlow Identity

B.3 关于 MeanFlow 恒等式的充分性

In the main paper, starting from the definitional Eq. (4), we have derived the MeanFlow Identity Eq. (6). This indicates that ”Eq. (4) $\Rightarrow$ Eq. (6)”, that is, Eq. (6) is a necessary condition for Eq. (4) Next, we show that it is also a sufficient condition, that is,”Eq. (6) $\Rightarrow$ Eq. (4)”.

在主文中，从定义式 Eq. (4) 出发，我们推导出了 MeanFlow 恒等式 Eq. (6)。这表明“Eq. (4) ⇒ Eq. (6)”，即 Eq. (6) 是 Eq. (4) 的必要条件。接下来，我们证明它也是充分条件，即“Eq. (6) ⇒ Eq. (4)”。

In general, equality of derivatives does not imply equality of integrals: they may differ by a constant. In our case, we show that the constant is canceled out. Consider a ”displacement field” S written as:

一般来说，导数相等并不意味着积分相等: 它们可能相差一个常数。在我们的情况下，我们证明该常数被抵消。考虑一个“位移场” S ，其表达为:

$$S\left(z_t, r, t\right)=(t-r)\, u\left(z_t, r, t\right). \tag{23}$$

If we treat S as an arbitrary function, then in general, equality of derivatives can only lead to equality of integrals, up to some constants:

如果将 S 视为任意函数，则一般情况下，导数相等只能导致积分相等，最多相差某个常数:

$$\frac{d}{dt} S\left(z_t, r, t\right)=v\left(z_t, t\right) \Rightarrow S\left(z_t, r, t\right)+C_1=\int_r^t v\left(z_\tau, \tau\right) d\tau+C_2. \tag{24}$$

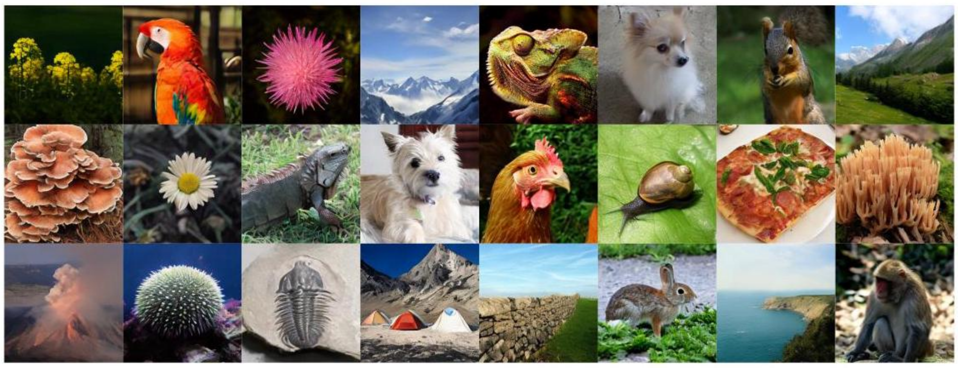


Figure 5: 1-NFE Generation Results. We show curated examples of class-conditional generation on ImageNet 256 ×256 × 256 using our 1-NFE model (MeanFlow-XL/2,3.43 FID).

图 5:1-NFE 生成结果。我们展示了使用我们的 1-NFE 模型 (MeanFlow-XL/2，3.43 FID) 在 ImageNet 256 ×256 × 256 上的类别条件生成精选示例。

However, the definition of S gives $S|_{t=r}=0$, and we also have $\int_r^t v d\tau=0$ when $t=r$, which gives $C_1=C_2$. This indicates ”Eq. (6) ⇒ Eq. (4)”.

然而， S gives $S|_{t=r}=0$ 的定义，以及当 $t=r$ 时我们还有 $\int_r^t v d\tau=0$ ，这给出了 $C_1=C_2$ 。这表明“Eq. (6) ⇒ Eq. (4)”。

We note that this sufficiency is a consequence of modeling the average velocity u , rather than directly the displacement S . Enforcing the equality of derivatives on the displacement, for example, $\frac{d}{dt} S=v$, does not automatically yield $S=\int_r^t v d\tau$. If we were to parameterize S directly, an extra boundary condition $S|_{t=r}=0$ is needed. Our formulation can automatically satisfy this condition.

我们注意到，这种充分性是由于对平均速度 u 的建模，而非直接对位移 S 的建模。对位移强制导数相等，例如 $\frac{d}{dt}S = v$ ，并不自动导致 $S = \int_r^t v d\tau$ 。如果直接参数化 S ，则需要额外的边界条件 $S|_{t=r} = 0$ 。我们的形式化方法可以自动满足该条件。

B.4 Analysis on Jacobian-Vector Product (JVP) Computation

B.4 关于 Jacobian-向量积 (JVP) 计算的分析

While JVP computation can be seen as a concern in some methods, it is very lightweight in ours. In our case, as the computed product (between the Jacobian matrix and the tangent vector) is subject to stop-gradient (Eq. (10)), it is treated as a constant when conducting SGD using backpropagation w.r.t. θ . Consequently, it does not add any overhead to the θ -backpropagation.

虽然 JVP 计算在某些方法中可能是个问题，但在我们的方法中非常轻量。因为计算的乘积 (Jacobian 矩阵与切向量的乘积) 受停止梯度操作 (Eq. (10)) 约束，在使用反向传播进行随机梯度下降时，它被视为常数，针对 θ 。因此，它不会增加 θ 反向传播的任何开销。

As such, the extra computation incurred by computing this product is the ”backward” pass performed by the jvp operation. This backward is analogous to the standard backward pass used for optimizing neural networks (w.r.t. θ)—in fact, in some deep learning libraries, such as JAX [4], they use similar interfaces to compute the standard backpropagation (w.r.t. θ). In our case, it is even less costly than a standard backpropagation, as it only needs to backpropagate to the input variables, not to the parameters θ . This overhead is small.

因此，计算该乘积所产生的额外计算量即为由 jvp 操作执行的“反向”过程。该反向过程类似于用于优化神经网络的标准反向传播 (相对于 θ)——实际上，在一些深度学习库中，如 JAX [4]，它们使用类似的接口来计算标准反向传播 (相对于 θ)。在我们的情况下，这一过程的开销甚至低于标准反向传播，因为它只需对输入变量进行反向传播，而不涉及参数 θ 。这一开销很小。

We benchmark the overhead caused by JVP in our ablation model, B/4. We implement the model in JAX and benchmark in v4-8 TPUs. We compare MeanFlow with its Flow Matching counterpart, where the only overhead is the backward pass incurred by JVP; notice that the forward pass of JVP is the standard forward propagation, which Flow Matching (or any typical objective function) also needs to perform. In our benchmarking, the training cost is 0.045 sec/iter using Flow Matching, and 0.052 sec/iter using MeanFlow, which is a merely 16% wall-clock overhead.

我们在消融模型 B/4 中对 JVP 引起的开销进行了基准测试。我们在 JAX 中实现该模型，并在 v4-8 TPU 上进行基准测试。我们将 MeanFlow 与其 Flow Matching 对应方法进行了比较，其中唯一的开销是 JVP 引起的反向过程；注意，JVP 的前向过程是标准的前向传播，而 Flow Matching(或任何典型的目标函数) 也需要执行该过程。在我们的基准测试中，使用 Flow Matching 的训练时间为 0.045 秒/迭代，使用 MeanFlow 为 0.052 秒/迭代，墙钟时间开销仅为 16%。

C Qualitative Results

C 定性结果

Fig. 5 shows curated generation examples on ImageNet 256 $\times$256 using our 1-NFE model.

图 5 展示了我们使用 1-NFE 模型在 ImageNet 256 $\times$256 上的精选生成示例。