视觉与语言中的常微分方程

肖桐 XIAOTONG@MAIL.NEU.EDU.CN
东北大学，中国

阮俊豪 RANGEHOW@OUTLOOK.COM
东北大学，中国

李北 LIBEI_NEU@OUTLOOK.COM
东北大学，中国

余正涛 ZTYU@HOTMAIL.COM
昆明理工大学，中国

张民 ZHANGMIN2021@HIT.EDU.CN
哈尔滨工业大学（深圳），中国

朱靖波 ZHUJINGBO@MAIL.NEU.EDU.CN
东北大学，中国

Abstract

常微分方程（ODEs）为理解复杂系统的连续演化提供了数学基础。尽管深度学习传统上依赖于离散的逐层计算，但视觉和语言领域中的许多最先进模型，从根本上可以被解释为连续动力学过程的离散化近似。在本文中，我们提出了一个统一的理论框架，通过 ODE 的形式体系，将多样化的神经架构和生成过程联系起来。我们证明，这种连续时间视角为模型设计提供了跨越三个递进维度的原则性方法。首先，我们展示了经典的残差网络和 Transformer 如何被表述为底层 ODE 的特定离散化方案，从而为先进架构的设计提供了动机。其次，我们分析了神经 ODE 和基于流的模型，并讨论了完全连续时间公式如何实现更灵活的密度估计和参数效率。第三，我们将扩散模型视为连接离散与连续框架的桥梁。我们表明，现代视觉和语言生成模型可以从微分方程的这一统一视角进行解释。更广泛地说，ODE 可以作为一种通用范式，用于提出和分析视觉与语言领域的深度学习问题。反过来，通过利用连续动力系统的强大数学工具包，这一视角可以驱动新的架构和算法设计。

目录

1. 数学预备知识

常微分方程（ODEs）的概念源于一个更广泛的概念——**微分方程**。这一领域研究变量如何变化的问题，并拥有悠久而成功的历史 [Edwards, 1994; Dunham, 2005]。如今，微分方程已成为现代科学与工程的基本语言，其应用范围非常广泛。在本节中，我们将介绍微分方程的基本概念。特别地，我们将重点讨论常微分方程，并展示这些数学工具在深度学习中的一些应用。

1.1 常微分方程表示法

我们首先引入一个移动汽车的例子作为本节贯穿始终的示例，用以激发几个关键概念。假设我们观察到一辆汽车沿直线行驶。令 $t \in \mathbb{R}$ 表示时间，它作为我们的自变量。我们将汽车在任意给定时间 t 的位置记为 $x(t)$。这里 $x(t)$ 是因变量，其值依赖于自变量 t。在某些情况下，为了简化符号，我们可能会省略参数（即使用 x 代替 $x(t)$）。

我们知道，汽车的速度是其位置相对于时间的变化率。在微积分中，这个量被表示为导数 $\frac{dx(t)}{dt}$。这里 $dx(t)$ 和 dt 被称为**微分**，分别表示因变量和自变量的无穷小变化。导数 $\frac{dx(t)}{dt}$ 描述了位置 $x(t)$ 相对于时间 t 的瞬时变化率。因此，这个概念是在一个时间点上定义的，而不是在一个有限区间上。从数学上讲，$\frac{dx(t)}{dt}$ 可以写作差商的极限

$$\frac{dx(t)}{dt} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}. \tag{1}$$

关于导数的正式定义，读者可以在微积分教科书中找到更多细节 [Apostol, 1991; Spivak, 2006]。

如果汽车的速度由一条取决于其当前位置和时间的规则所支配（例如，由于道路摩擦或风阻的变化），我们可以使用一个函数 $f(\cdot)$ 来表达这种关系

$$\frac{dx(t)}{dt} = f(x(t), t). \tag{2}$$

这个方程在形式上被称为常微分方程。术语“常”意味着因变量 $x(t)$ 是单个自变量 t 的函数。这与**偏微分方程**（PDEs）形成对比，在偏微分方程中，未知函数依赖于多个自变量[1]。

或者，方程 (2) 也可以写成微分形式

$$dx(t) = f(x(t), t)dt. \tag{4}$$

这种表示法表明位置的无穷小变化是速度与时间增量的乘积。我们将在后续章节中展示，这种形式对于发展数值积分方法和建模随机过程特别有用。

显然，由方程 (2) 和 (4) 给出的常微分方程的复杂性取决于函数 $f(\cdot)$ 的复杂性。最简单的情况是 $f(\cdot)$ 为常数。例如，如果汽车以恒定速度 v 行驶，那么函数的形式为 $f(x(t), t) = v$。在这种

1. 偏微分方程的一个例子是**热传导方程**，它描述温度 u 如何随时间 t 和空间位置 s 变化。它通常写作

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial s^2}, \tag{3}$$

其中 α 是一个正系数。在这种情况下，方程涉及偏导数，用符号 ∂ 表示。$\frac{\partial^2 u}{\partial s^2}$ 是 u 在 s 方向上的二阶空间导数。

情况下，方程 (2) 简化为

$$\frac{dx(t)}{dt} = v. \tag{5}$$

很容易看出，这个常微分方程对应于我们熟悉的线性运动方程

$$x(t) \quad = \quad vt + x(0), \tag{6}$$

其中 $x(0)$ 表示汽车在时间 $t = 0$ 时的位置。这个简单的例子展示了微分方程的一个基本性质：通解通常包含一个任意常数（此处为 $x(0)$）。从几何上讲，这意味着常微分方程描述了 (x, t) 平面上一族平行的曲线，每条曲线对应一个不同的起点。

当 $f(\cdot)$ 明确依赖于位置 $x(t)$ 时，情况变得更有趣。例如，假设由于某种控制规则，汽车的速度随着其远离原点而减小。如果函数 $f(\cdot)$ 线性依赖于 $x(t)$（例如，对于某个常数 k，$f(x(t), t) = -kx(t)$），则该方程被归类为**线性常微分方程**。这类方程特别受关注，因为它们通常允许闭式解析解 [Hartman, 2002]。

然而，在许多实际应用中，$f(\cdot)$ 是非线性的。如果我们的汽车在一个其支配关系涉及诸如 $x(t)^2$ 或 $\sin(x(t))$ 等项的环境中运动，我们将处理一个**非线性常微分方程**。与线性方程不同，非线性方程很少具有简单的解析解。因此，精确的解析推导通常是不可能的，我们必须求助于数值近似技术。由于后续章节讨论的大多数问题都涉及非线性常微分方程，探索近似解将是接下来讨论的重点。

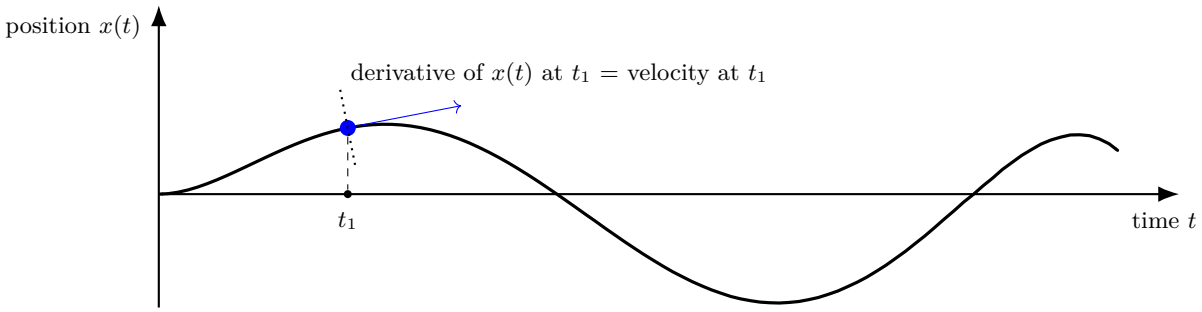
请注意，尽管 $f(\cdot)$ 接受两个参数 $x(t)$ 和 t，但项 $x(t)$ 本身是 t 的函数。因此，沿着任何特定的轨迹，速度可以被看作是 t 的复合函数。在这种情况下，我们可以将 $f(\cdot)$ 绘制为 t 的函数，其中任意点的值对应于导数 $\frac{dx(t)}{dt}$。参见图 1 获取一个简单示例。

然而，在实践中，$f(\cdot)$ 对 t 的具体依赖性通常是未知的，因为轨迹 $x(t)$ 本身是未知的。此外，$f(\cdot)$ 对时间的显式依赖性带来了额外的复杂性。与具有静态结构的自治函数不同，这里的函数形式本身随时间演变（为了强调这一点，我们可以采用符号 $f_t(\cdot)$ 来表示 $f(\cdot, t)$）。这意味着即使汽车在稍后的时刻回到完全相同的位置，它也可能具有不同的速度。由于“道路规则”不是恒定的，汽车的行为不能仅由其当前状态 $x(t)$ 预测，还必须考虑特定的时间点。

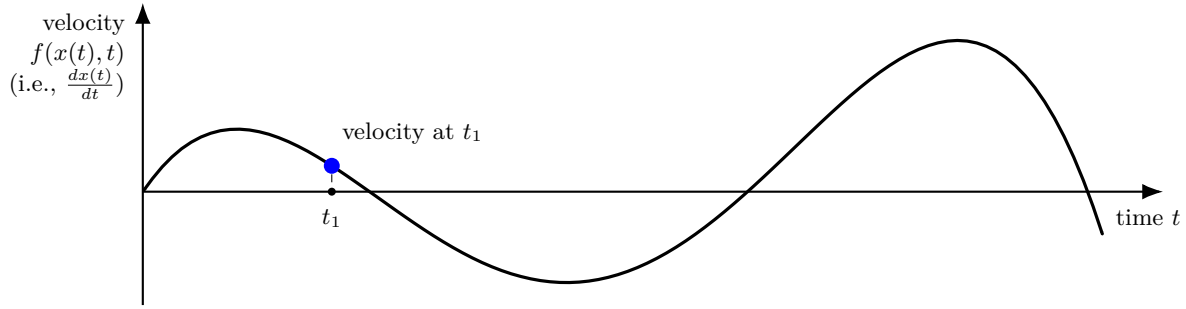
更根本地说，这样的常微分方程定义了一个**动力系统** [Hirsch et al., 2013]。在这个框架中，变量 $x(t)$ 表示系统的**状态**，它描述了系统在任何特定时刻的配置。函数 $f(\cdot)$ 表征了系统的**动力学**（通常称为**动态规则**），它指示了状态在任意给定时刻如何变化。从这个角度来看，关于时间依赖性的区别成为规则本身的分类：如果动态规则是静态的且仅依赖于状态（即 $\frac{dx(t)}{dt} = f(x(t))$），则系统是**自治的**。如果规则本身随时间变化（即 $\frac{dx(t)}{dt} = f(x(t), t)$），则系统是**非自治的**。非自治系统是描述复杂过程的强大工具，尽管它们本质上比自治系统更复杂。例如，在第 2 节中，我们将展示 Transformer 层堆栈可以建模为一个非自治系统。

到目前为止，我们的讨论集中在标量变量和函数上，即状态 $x(t)$ 是标量变量，函数 $f(\cdot)$ 是标量函数。一个自然的扩展是在向量上定义常微分方程。令 $\mathbf{x}(t) \in \mathbb{R}^d$ 表示一个包含 d 个不同分量的向量状态（例如，神经网络层中的神经元激活）。相应地，函数 $f(\cdot)$ 被定义为一个向量值映射：$\mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$。那么，常微分方程变为

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t). \tag{7}$$



(a) Position $x(t)$ against time t



(b) Velocity $f(x(t), t)$ (i.e., $\frac{dx(t)}{dt}$) against time t

图 1: 常微分方程因变量 ($x(t)$) 及其导数 ($\frac{dx(t)}{dt}$) 的曲线图。虽然该微分方程需要将 $x(t)$ 作为 $f(\cdot)$ 的输入，但 $x(t)$ 本身由时间驱动。因此，若轨迹固定，则速度 $f(\cdot)$ 隐式地成为仅关于 t 的函数。故可直接绘制速度 $f(\cdot)$ 与 t 的关系曲线。

这里，导数 $\frac{d\mathbf{x}(t)}{dt}$ 是逐分量地应用于向量 $\mathbf{x}(t)$ 的。这种向高维度的转换引入了更丰富的几何解释。虽然我们先前示例中的标量函数仅表示曲线的斜率，但向量函数 $f(\cdot)$ 在状态空间上定义了一个**向量场**。在每一点 $\mathbf{x}(t)$ 处，$f(\mathbf{x}(t), t)$ 分配一个向量，该向量指定了系统瞬时速度的大小和方向。

尽管我们将状态扩展到 d 维空间，但常微分方程的基本概念和性质仍然有效。在本节中，为简单起见，我们将继续使用标量表示法。然而，在后续章节中，我们将主要使用向量值常微分方程来建模更复杂的问题。我们在表 1 中总结了迄今为止引入的关键概念。

1.2 求解常微分方程

我们已经说明常微分方程描述了状态 $x(t)$ 的导数。有人可能会问：为什么我们使用导数来定义系统，而不是直接描述状态 $x(t)$？确实，如果 $x(t)$ 的完整轨迹已知，那么常微分方程将是多余的。然而，在许多应用中，我们并没有 $x(t)$ 的解析表达式。相反，我们只被提供了动力学 $f(\cdot)$ 和初始条件 $x(0)$。这类似于驾驶汽车：虽然在任何时刻都很容易知道我们当前的速度，但要确定相对于起点的当前位置则要困难得多。用常微分方程的语言来说，我们被给予了“速度计” $f(\cdot)$ 和起点 $x(0)$，我们的目标是重建旅程的“地图” $x(t)$。

Notation	描述
t	自变量，通常表示时间。
$x(t)$	因变量，表示系统在 时间 t 时的状态。
$dx(t)/dt$	状态的导数，表示瞬时 变化率（例如，速度）。
$f(x(t), t)$	控制系统 演化的函数（或动力学规则）。它描述了系统的动力学。
$\frac{dx(t)}{dt} = f(x(t), t)$	非自治常微分方程，其中动力学显式 依赖于时间。
$\frac{dx(t)}{dt} = f(x(t))$	自治常微分方程，其中动力学仅 依赖于状态（时间不变的规则）。
$dx(t) = f(x(t), t)dt$	常微分方程的微分形式，通常用于 数值积分和随机微积分。
$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t)$	向量值常微分方程，描述具有 多维状态向量 $\mathbf{x}(t) \in \mathbb{R}^d$ 的系统（其中 $f(\cdot)$ 定义一个向量场）。

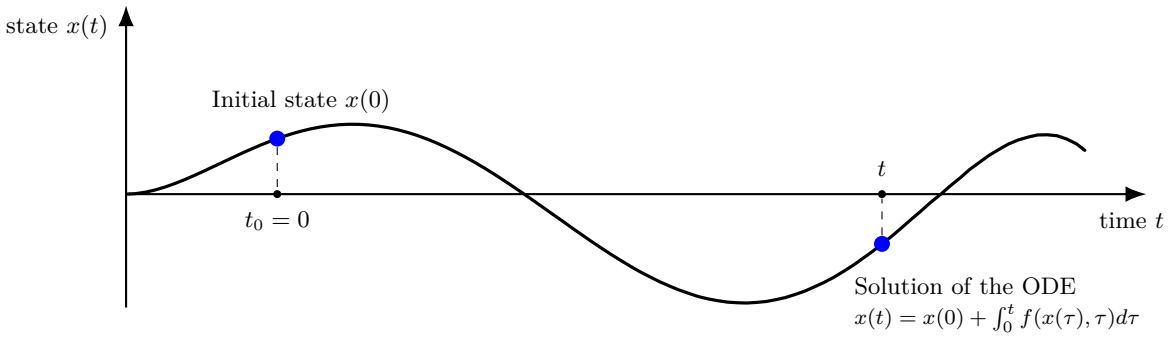
表 1: 常微分方程中的基本概念。

在这种情况下，我们需要求解常微分方程。这通常被称为**初值问题**，其中给定一个常微分方程和一个特定的约束条件（例如初始状态 $x(0)$），我们从其导数中恢复任意时间 t 的状态 $x(t)$。根据微积分基本定理，时间 t 的状态是初始状态与区间 $[0, t]$ 上所有瞬时变化累积的总和。因此，我们可以通过积分写出常微分方程的解

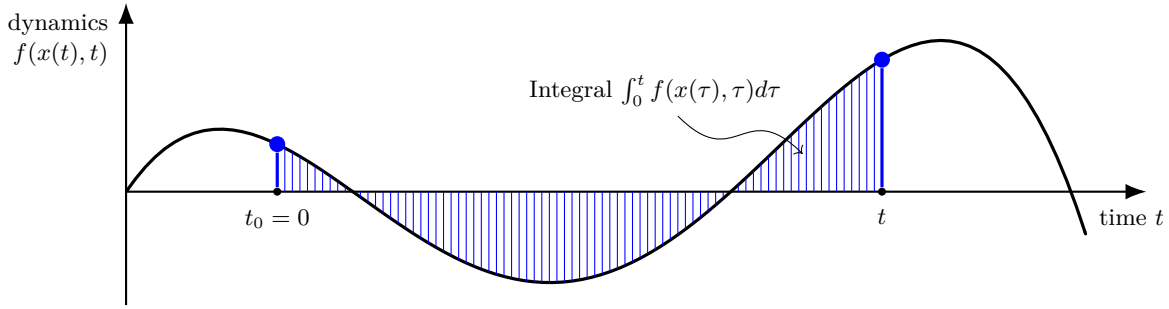
$$\begin{aligned} x(t) &= x(0) + \int_0^t \frac{dx(\tau)}{d\tau} d\tau \\ &= x(0) + \int_0^t f(x(\tau), \tau) d\tau. \end{aligned} \tag{8}$$

如图 2 所示，这个公式有一个非常直观的几何解释。积分项 $\int_0^t f(x(\tau), \tau)d\tau$ 表示函数 $f(x(\tau), \tau)$ 曲线下从 $\tau = 0$ 到 $\tau = t$ 的带符号面积。这个面积量化了状态 $x(t)$ 在区间 $[0, t]$ 上的总累积变化。因此，该方程简单地表明，时间 t 的状态是通过将这个总变化（面积）加到初始状态 $x(0)$ 上得到的。

然而，这种积分形式提出了常微分方程求解的一个计算挑战：虽然方程看起来简单，但它无法直接计算，因为被积函数 $f(x(\tau), \tau)$ 依赖于未知的状态轨迹 $x(\tau)$ 本身。此外，即使轨迹是隐式定义的，由于动力学 $f(\cdot)$ 的复杂性和非线性（例如，当使用深度神经网络建模时），解析地计算积分通常也很困难。相反，我们必须借助数值方法来近似积分。这些方法的核心思想是**离散化**：与其跟踪 $x(t)$ 在每一个无穷小瞬间的连续演化，我们在一系列离散时间步 $\{t_0, t_1, \ldots, t_N\}$ 上近似轨迹。由于在有限个点上计算函数 $f(\cdot)$ 在计算上是廉价的，离散化提供了一种非常有效的方式来获得近似解。



(a) State $x(t)$ against time t



(b) Dynamics $f(x(t), t)$ against time t

图 2: 通过积分求解常微分方程 $\frac{dx(t)}{dt} = f(x(t), t)$ 的示意图。蓝色阴影区域表示积分 $\int_0^t f(x(\tau), \tau) d\tau$。在时间 t 处的常微分方程解由初始状态 $x(0)$ 与该积分之和给出：$x(t) = x(0) + \int_0^t f(x(\tau), \tau) d\tau$。

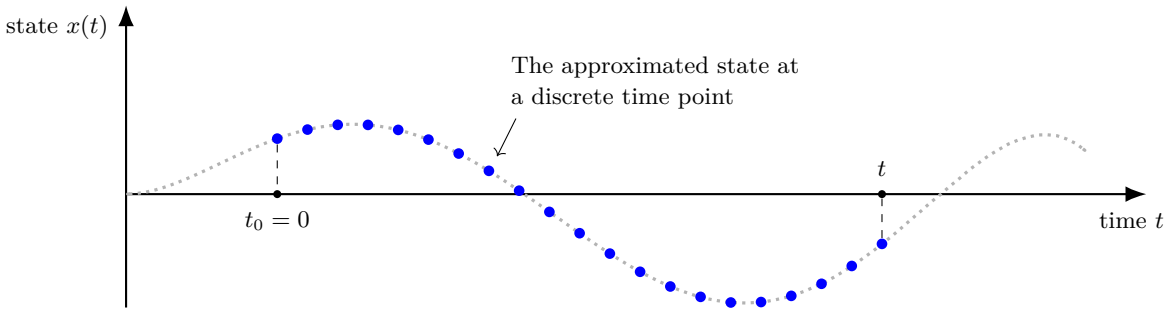
离散化可以从黎曼积分的角度来理解。我们可以通过将积分区间划分为小的子区间，并对由导数函数定义的矩形面积求和，来近似方程 (8) 中的连续积分。图 3 展示了一个示意图。为了形式化这一点，让我们在区间 $[0, t]$ 上定义一个由 $N + 1$ 个点 $t_0 < t_1 < \cdots < t_N$ 组成的时间网格，其中 $t_0 = 0$ 是开始时间，$t_N = t$ 是结束时间。为简单起见，我们通常假设时间间隔是恒定的，即

$$\Delta t = t_{k+1} - t_k, \tag{9}$$

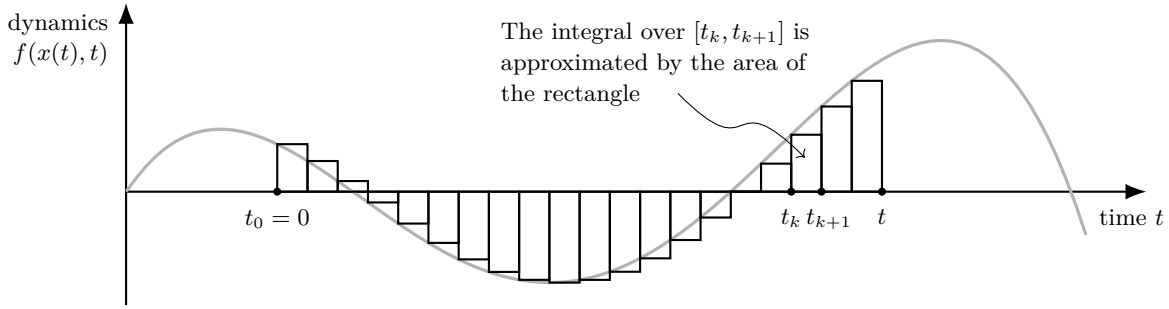
其中 Δt 被称为**步长**。

我们的目标是计算一个值序列 $\{\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_N\}$，其中每个 $\bar{x}_k$ 作为真实状态 $x(t_k)$ 的近似。从当前状态 $\bar{x}_k$ 到下一个状态 $\bar{x}_{k+1}$ 的转移需要近似动力学在小区间 $[t_k, t_{k+1}]$ 上的积分。数学上，我们可以使用以下递归形式表达离散化模型

$$\bar{x}_{k+1} = \bar{x}_k + \text{Approx}\left(\int_{t_k}^{t_{k+1}} f(x(\tau), \tau) d\tau \right), \tag{10}$$



(a) State $x(t)$ against time t



(b) Dynamics $f(x(t), t)$ against time t

图 3: 常微分方程离散化示意图。函数 $f(\cdot)$ 在多个离散时间点上被求值。在每个时间点处，获得一个状态的近似值，用蓝色圆圈表示。$f(\cdot)$ 的积分决定了状态的演化，它在每个子区间上的近似值由一个矩形的面积给出，即 $\Delta t \cdot \Psi(\bar{x}_k, t_k, \Delta t, f)$，其中 Δt 是宽度，$\Psi(\bar{x}_k, t_k, \Delta t, f)$ 是高度。

其中函数 Approx$(\cdot)$ 近似局部积分。在离散化的常微分方程模型中，这个函数通常使用 $f(\cdot)$ 的局部几何来定义，像这样

$$\text{Approx}\left(\int_{t_k}^{t_{k+1}} f(x(\tau), \tau) d\tau\right) \quad = \quad \Delta t \cdot \Psi(\bar{x}_k, t_k, \Delta t, f). \tag{11}$$

那么，方程 (10) 可以重写为

$$\bar{x}_{k+1} = \bar{x}_k + \Delta t \cdot \Psi(\bar{x}_k, t_k, \Delta t, f). \tag{12}$$

这里，$\Psi(\cdot)$ 被称为**增量函数**或**步进函数**。它使用在可用离散点上计算的 $f(\cdot)$ 值来估计区间上的积分。不同的数值方法对应于 $\Psi(\cdot)$ 的不同选择。例如，基于左端点 (t_k) 斜率选择 $\Psi(\cdot)$ 对应于经典的**欧拉方法**（类似于左黎曼和）。在这种情况下，增量函数就是动力学函数本身：$\Psi(\bar{x}_k, t_k, \Delta t, f) =$

$f(\bar{x}_k, t_k)$。将其代回方程 (12) 得到欧拉方法的更新规则

$$\bar{x}_{k+1} = \bar{x}_k + \Delta t \cdot f(\bar{x}_k, t_k). \tag{13}$$

由于每步只需要一次函数求值，欧拉方法在计算上是高效的。然而，它通常精度较差，特别是当步长 Δt 较大或底层动力学 $f(\cdot)$ 变化剧烈时。为了解决这些限制，数值求解器通常遵循两个主要策略：高阶单步方法和多步方法。

- **高阶单步方法**。在数值分析中，方法的阶数是指当步长 Δt 趋近于零时近似误差的收敛速率。若局部误差按 $O((\Delta t)^{p+1})$ 变化，则称该方法具有 p 阶精度。经典欧拉法为一阶方法，而高阶方法通过评估区间 $[t_k, t_{k+1}]$ 内多个中间点处的动力学函数 $f(\cdot)$ 来更精确地估计平均斜率。例如，四阶**龙格-库塔方法**（RK4）通过计算四个不同斜率（起点一个、中点两个、终点一个）的加权平均值，能够在无需极细步长的条件下实现高精度计算。

- **多步方法**。与仅依赖当前状态 $\bar{x}_k$ 计算下一状态 $\bar{x}_{k+1}$ 的单步方法（如欧拉法和 RK4）不同，多步方法充分利用轨迹历史信息。通过将过去导数值拟合为多项式，诸如 Adams-Bashforth 系列的方法可以预测下一区间的积分值。多步方法的主要优势在于计算效率：它们通过复用先前计算的 $f(\cdot)$ 函数值（而非每步执行多次新的昂贵函数评估）即可实现高阶精度。但这类方法不具备自启动特性，需借助单步方法完成轨迹前几步的初始化。

还有其他方法可以改进常微分方程求解器。例如，在**自适应步长方法**中，恒定步长 Δt 的假设被放宽。相反，求解器在每一步动态调整 Δt。这里我们不深入探讨这些常微分方程求解器的细节，在接下来的讨论中将它们作为标准工具使用。感兴趣的读者可以参考数值分析书籍以获取更多细节 [Stoer et al., 1980; Burden et al., 2015]。

在继续之前，有必要澄清本工作中使用的符号，因为文献中的惯例各不相同。通常，函数符号 $x(t)$ 表示连续时间状态变量。相比之下，下标符号 x_t 在机器学习和随机微积分中经常用于表示特定时间快照 t 的状态。在数值求解器的上下文中，x_k（或 x_{t_k}）通常指对应于时间 t_k 的第 k 个积分步的计算值。

在我们之前的讨论中，我们区分了数值近似 $\bar{x}_k$ 和真实解析解 $x(t_k)$。然而，为了避免符号混乱，并使我们的数学表示与现有文献更加一致，我们将使用 x_k（不带横杠）来表示数值方法中第 k 步的估计值。更一般地，我们可以使用 x_k 来表示某个离散时间步的状态。虽然这引入了一点符号滥用，但其含义从上下文中是清晰的。例如，欧拉方法可以写成

$$x_{k+1} = x_k + \Delta t \cdot f(x_k, t_k) \tag{14}$$

同样地，偶尔我们可能会使用 $x(t)$ 而不是 $\bar{x}(t)$ 来表示时间 t 的状态估计值，其中与真实解的区别并不关键。

总而言之，给定一个由动力学 f 支配的常微分方程和一个初始状态 $x(0)$，我们可以将常微分方程求解器视为一个函数 $\phi(\cdot)$。对于任何 $t \geq 0$，该函数返回状态的近似值，记为

$$x(t) \quad = \quad \phi(f, x(0), t), \tag{15}$$

同时满足初始条件 $x(0) = \phi(f, x(0), 0)$。

1.3 两个示例

现在，我们通过两个示例来说明常微分方程的应用。第一个示例涉及训练中参数更新的建模，第二个示例则关注**连续时间马尔可夫链**。

1.3.1 使用梯度下降进行训练

训练神经网络最流行的方法是梯度下降。其思想是，我们首先定义一个损失函数 $L(\theta)$ 来量化模型预测与真实值之间的差异，其中 $\theta \in \mathbb{R}^d$ 表示模型参数向量。为了最小化这个损失，我们沿着最陡下降方向迭代调整参数，该方向由负梯度 $-\frac{dL(\theta)}{d\theta}$ 给出。在每次迭代 k 中，更新规则由下式给出：

$$\theta_{k+1} = \theta_k - \eta \cdot \frac{dL(\theta_k)}{d\theta_k}, \tag{16}$$

其中 $\eta > 0$ 是学习率。

回顾方程（14），求解常微分方程的欧拉方法由 $x_{k+1} = x_k + \Delta t \cdot f(x_k, t_k)$ 给出。通过比较这两个方程，我们可以观察到它们之间的直接对应关系：

$$\begin{aligned} \text{State } x_k &\quad\Leftrightarrow\quad \text{Parameters } \theta_k \\ \text{Step Size } \Delta t &\quad\Leftrightarrow\quad \text{Learning Rate } \eta \\ \text{Dynamics } f(x_k, t_k) &\quad\Leftrightarrow\quad \text{Negative Gradient } -dL(\theta_k)/d\theta_k \end{aligned}$$

考虑到这些方程的相似性，我们可以将梯度下降视为一个连续时间常微分方程的显式欧拉离散化。如果我们取极限，让学习率 $\eta \to 0$，相应地步数 $k \to \infty$，则离散参数更新的序列收敛于以下常微分方程的解：

$$\frac{d\theta_t}{dt} = -\frac{dL(\theta_t)}{d\theta_t}. \tag{17}$$

这个常微分方程在形式上被称为**梯度流**。想象一个无质量的粒子在一个复杂的表面（例如，损失景观）上移动。由于粒子没有质量，其瞬时速度完全由地形的斜率决定。斜率越陡，粒子移动得越快，并严格遵循负梯度的方向。这种连续的下降过程正是方程（17）所描述的。

通过将梯度下降与常微分方程联系起来，我们可以利用成熟的常微分方程理论来更好地理解神经网络优化的行为。例如，常微分方程解的稳定性可以解释为什么学习率过大时训练可能发散（在数值分析中称为刚性问题）。此外，这种视角可以自然地扩展到更高级的优化器。考虑带有动量的随机梯度下降 [Qian, 1999]，它旨在通过累积过去的梯度来加速训练。其更新规则通常表述为一个由两个耦合方程组成的系统：

$$v_{k+1} = \mu v_k - \eta \cdot \frac{dL(\theta_k)}{d\theta_k} \tag{18}$$

$$\theta_{k+1} = \theta_k + v_{k+1}, \tag{19}$$

其中 v_k 表示第 k 步的动量变量，而 $\mu \in [0, 1)$ 是动量系数，用于控制保留多少过去的累积量。

如果我们分析这些离散更新的连续时间极限（通过令步长 $\eta \to 0$），我们将不再得到一个一阶常微分方程。相反，该系统收敛于一个二阶常微分方程 [Su et al., 2016]：

$$\frac{d^2\theta_t}{dt^2} + \lambda \frac{d\theta_t}{dt} = -\frac{dL(\theta_t)}{d\theta_t}, \tag{20}$$

其中 λ 是一个与阻尼摩擦相关的系数。这个方程在物理上类似于牛顿第二运动定律，在形式上被称为**重球常微分方程**。感兴趣的读者可以参考附录 A 以获取关于重球常微分方程更详细的推导。

1.3.2 连续时间马尔可夫链

连续时间马尔可夫链（CTMC）描述了一个在有限个离散状态之间转移的随机过程，其中转移在时间上是连续发生的 [Norris, 1998]。虽然 CTMC 通常通过跳跃过程引入，但它们也可以被建模为一个支配概率分布演化的确定性线性动力系统。这一视角将概率论与常微分方程联系起来。

考虑一个具有 N 个离散状态的系统，状态集记为 $\mathcal{S} = \{1, 2, \ldots, N\}$。我们不追踪单个随机轨迹 $x(t)$，而是关注所有状态上概率分布的演化。令 $\mathbf{p}(t) = \begin{bmatrix} p_1(t) & p_2(t) & \cdots & p_N(t) \end{bmatrix}^\top \in \mathbb{R}^N$ 为一个列向量，其第 i 个分量表示在时刻 t 处于状态 i 的概率：

$$p_i(t) \quad = \quad \Pr(x(t) = i), \tag{21}$$

满足约束条件

$$\sum_{i=1}^{N} p_i(t) \quad = \quad 1. \tag{22}$$

为了描述系统的演化，我们需要指定这个概率分布如何随时间变化。直观上，我们可以将其视为状态之间的概率质量流动。这种流动的动力学由转移速率矩阵 $\mathbf{Q} \in \mathbb{R}^{N \times N}$ 定义，该矩阵通常被称为无穷小生成元。$\mathbf{Q}$ 的元素量化了状态间概率质量的流动速率：

- 非对角元素（q_{ij}，其中 $i \neq j$）：表示从状态 i 到状态 j 的瞬时转移速率。它们必须是非负的（$q_{ij} \geq 0$）。
- 对角元素（q_{ii}）：定义为 $q_{ii} = -\sum_{j \neq i} q_{ij}$。这确保了每一行的和为零，反映了总概率的守恒（流出的总量必须等于离开的总速率）。

给定无穷小生成元，概率向量 $\mathbf{p}(t)$ 的演化可以由一个线性常微分方程组定义，称为 **柯尔莫哥洛夫向前方程**（或在物理学中称为**主方程**）：

$$\frac{d\mathbf{p}(t)}{dt} \quad = \quad \mathbf{Q}^\top \mathbf{p}(t). \tag{23}$$

该常微分方程代表了概率流的全局平衡。矩阵乘积 $\mathbf{Q}^\top \mathbf{p}(t)$ 汇总了每个状态的概率流。具体来说，对于状态 j，变化率 $\frac{dp_j}{dt}$ 由从所有其他状态进入 j 的总概率质量（流入）与离开 j 的质量（流出）之差决定。因此，方程 (23) 简单地表明，概率分布的变化率线性依赖于当前分布和转移速率。

如果 $\mathbf{Q}$ 是常数，则 CTMC 是时间齐次的。在这种情况下，方程 (23) 是一个标准的线性常微分方程组，其解析解由矩阵指数给出：

$$\mathbf{p}(t) = e^{\mathbf{Q}^\top t}\mathbf{p}(0), \tag{24}$$

其中 $\mathbf{p}(0)$ 是初始分布，$e^{\mathbf{Q}^\top t}$ 是将初始分布映射到时刻 t 分布的传播子。在第 5 节中，我们将展示 CTMC 可以作为离散扩散模型的基础。

2. Transformer 的常微分方程视角

在上一节中，我们看到，虽然常微分方程最初是为建模连续时间过程而设计的，但它们能够通过离散化来处理离散时间问题。事实上，根据所采用的离散化方案，我们可以将常微分方程应用于不同的问题。在本节中，我们将讨论一个重要的案例，即常微分方程被完全离散化以启发神经网络设计。具体来说，我们从离散化常微分方程的视角，阐述对基于残差网络的模型（如 Transformer）的解释与改进。我们表明，常微分方程可以为设计此类离散结构提供洞见。

2.1 Transformers 与常微分方程的联系

在此，我们将 Transformer 视为序列模型，它接收一个词元序列作为输入，并输出上下文相关的表示 [Vaswani et al., 2017]。例如，大多数最新的**大语言模型**（LLM）都基于 Transformer 解码器架构 [Brown et al., 2020; Bai et al., 2023; Liu et al., 2024]。该架构由一系列相同的层堆叠而成，称为 Transformer 层。每个 Transformer 层内包含两个子层：一个**多头自注意力**（MHSA）子层和一个位置级的**前馈网络**（FFN）子层。Transformer 中的一个关键设计选择是使用残差连接 [He et al., 2016]，它引入了一条捷径路径，将输入直接加到子层的输出上。

在这项工作中，我们考虑**预归一化**架构，该架构将层归一化应用于子层的输入，而不是在残差求和之后 [Wang et al., 2019; Baevski and Auli, 2019]。这种架构在最新的 LLM 中被广泛使用，因为它有助于训练非常深的网络。形式上，让我们将网络视为一系列残差块（即子层）。令 $\mathbf{x}_l$ 表示第 l 个残差块的输入。该块的更新规则可以表示为

$$\mathbf{x}_{l+1} \quad = \quad \mathbf{x}_l + F(\mathrm{LN}(\mathbf{x}_l), \theta_l), \tag{25}$$

其中 $\mathbf{x}_l \in \mathbb{R}^d$ 表示输入，$\mathbf{x}_{l+1} \in \mathbb{R}^d$ 表示输出，$\mathrm{LN}(\cdot)$ 表示层归一化函数，$F(\cdot)$ 表示子层的变换函数（自注意力或 FFN），θ_l 表示函数 $F(\cdot)$ 的参数。

这里我们将 $\mathrm{LN}(\cdot)$ 吸收到 $F(\cdot)$ 中，从而使 $F(\cdot)$ 成为一个复合函数，先执行层归一化，再进行子层变换。然后，我们可以将公式 (25) 重写为

$$\mathbf{x}_{l+1} \quad = \quad \mathbf{x}_l + F(\mathbf{x}_l, \theta_l). \tag{26}$$

这种形式与前一小节讨论的欧拉方法非常相似。为了将 Transformer 残差块与常微分方程联系起来，让我们考虑一个描述状态 $\mathbf{x}(t)$ 如何随时间 t 变化的常微分方程

$$\frac{d\mathbf{x}(t)}{dt} \quad = \quad f(\mathbf{x}(t), \theta(t), t) \tag{27}$$

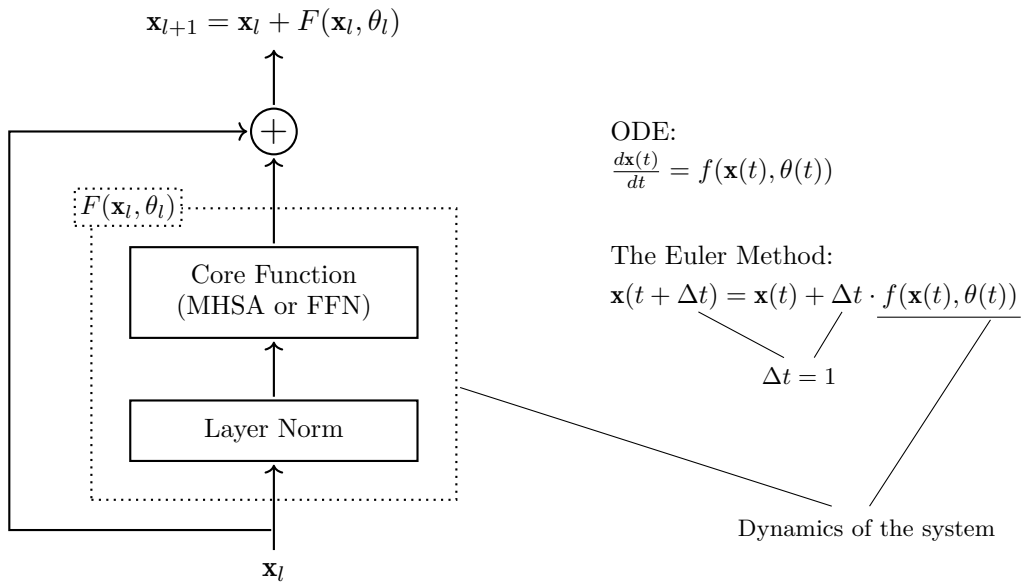


图 4: Transformer 子层及其对应 ODE 的示意图。该子层可描述为 $\mathbf{x}_{l+1} = \mathbf{x}_l + F(\mathrm{LN}(\mathbf{x}_l), \theta_l)$，其中 $\mathbf{x}_l$ 的加法表示残差连接，而 $F(\mathrm{LN}(\mathbf{x}_l), \theta_l)$ 表示由该子层定义的变换。这对应于 ODE $\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta(t))$ 的欧拉离散化。更具体地，对于更新规则 $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot f(\mathbf{x}(t), \theta(t))$，我们将 $f(\mathbf{x}(t), \theta(t))$ 视为子层的变换函数，并设 $\Delta t = 1$。

这是一个**非自治**常微分方程，因为其动力学通过时变参数 $\theta(t)$ 显式地依赖于时间。为了简化符号，我们将使用 $f(\mathbf{x}(t), \theta(t))$ 作为 $f(\mathbf{x}(t), \theta(t), t)$ 的简写。

给定离散时间步长 Δt，下一个时间步的状态可以通过欧拉离散化来近似

$$\mathbf{x}(t + \Delta t) \quad = \quad \mathbf{x}(t) + \Delta t \cdot f(\mathbf{x}(t), \theta(t)). \tag{28}$$

通过比较公式 (26) 和公式 (28)，我们可以在 Transformer 的残差连接与常微分方程的欧拉离散化之间建立直接对应关系。具体来说，如果我们设置时间步长 $\Delta t = 1$，那么 Transformer 子层的更新规则在数学上就等价于欧拉方法的一个单步。

在这个类比中，层索引 l 对应于离散时间步 t，隐藏状态 $\mathbf{x}_l$ 对应于状态 $\mathbf{x}(t)$。因此，变换函数 $F(\mathbf{x}_l, \theta_l)$ 充当了在常微分方程中控制动力学的导数函数 $f(\mathbf{x}(t), \theta(t))$。需要注意的是，虽然连续常微分方程公式中的 $\theta(t)$ 表示一个在时间 t 上连续定义的参数轨迹，但 Transformer 中不同的参数 θ_l 可以视为这个连续轨迹在整数时间步上的离散化值。图 4 说明了 Transformer 子层与常微分方程之间的对应关系。

因此，一个具有 L 层（即 $2L$ 个子层）的 Transformer 的前向传播，可以解释为使用单位步长的欧拉方法，从初始时间 $t = 0$ 到最终时间 $t = 2L$ 对一个常微分方程进行数值积分。这种观点表明，训练 Transformer 本质上是在学习一个动力系统的轨迹。在这个模型中，网络的深度不仅仅是一个结构超参数，还可以被视为一个动态过程的持续时间，该过程控制着输入表示的变换。这种见解也推动了基于连续时间常微分方程的神经模型的发展，例如神经常微分方程，这将在第 3 节中讨论。

2.2 ODE 启发的架构设计

尽管标准的预归一化 Transformer 可被解释为 ODE 的欧拉离散化，但从数值分析的角度看，它们存在局限性。例如，欧拉离散化是一种一阶方法，其局部截断误差为 $O((\Delta t)^2)$。在 Transformer 中，随着网络深度增加，误差可能会显著累积。因此，Transformer 计算出的轨迹可能会偏离底层 ODE 的理想动态。此外，标准的欧拉方法是一种显式方法，通常被认为在数值稳定性上不如隐式 ODE 求解器。更进一步，Transformer 通过顺序执行自注意力层和前馈层来更新表征，这会在建模耦合动态时引入分裂误差。因此，一些研究提出用更复杂的方案来替代简单的欧拉离散化，以设计更好的神经架构。

2.2.1 从物理系统角度解读架构

一种思路是借鉴其他领域中已建立良好常微分方程基础的替代性框架。例如，Lu et al. [2020] 将 Transformer 解释为多粒子动力学系统中对流-扩散方程的求解器。具体而言，考虑一个由多头自注意力子层和前馈网络子层构成的 Transformer 层，其表达式为：

$$\mathbf{x}_{l+1} = \mathbf{x}_l + F_{\text{att}}(\mathbf{x}_l, \omega_l) \tag{29}$$

$$\mathbf{x}_{l+2} = \mathbf{x}_{l+1} + F_{\text{ffn}}(\mathbf{x}_{l+1}, \pi_{l+1}) \tag{30}$$

其中 $F_{\text{att}}(\cdot)$ 和 $F_{\text{ffn}}(\cdot)$ 分别表示多头自注意力子层和前馈网络子层的函数，ω_l 和 π_{l+1} 为对应参数。将式 (29) 代入式 (30) 并引入层索引 k（令 $\mathbf{h}_k = \mathbf{x}_{2k}$），可得第 k 个 Transformer 层的统一表达式：

$$\mathbf{h}_{k+1} = \mathbf{h}_k + F_{\text{att}}(\mathbf{h}_k, \omega_k) + F_{\text{ffn}}\big(\mathbf{h}_k + F_{\text{att}}(\mathbf{h}_k, \omega_k), \pi_k\big) \tag{31}$$

从物理系统的视角看，该形式类似于对流-扩散方程，该方程同时模拟扩散过程与对流过程。定义连续算子 $\text{Att}(\mathbf{h}) = F_{\text{att}}(\mathbf{h}, \omega_k)$ 和 $\text{FFN}(\mathbf{h}) = F_{\text{ffn}}(\mathbf{h}, \pi_k)$。可将式 (31) 视为如下常微分方程的数值离散化：

$$\frac{d\mathbf{h}(t)}{dt} = \text{Att}(\mathbf{h}(t)) + \text{FFN}(\mathbf{h}(t)) \tag{32}$$

此常微分方程与本文先前提出的方程不同，因其右侧包含两个独立项。在物理学和数学中，此类常微分方程十分常见，例如物理系统在两种不同力同时作用下演化。直接求解此类组合方程通常较为困难或计算代价高昂。此时，常采用称为**算子分裂格式**的数值技术，通过交替独立求解各项的动力学来近似获得解。由此视角观之，标准 Transformer 的串行更新规则（式 (31)）本质上对应于通过欧拉离散化实现的 **Lie-Trotter 分裂格式**。该方法首先通过扩散动力学推进状态：

$$\frac{d\mathbf{h}(t)}{dt} = \text{Att}(\mathbf{h}(t)) \tag{33}$$

对初始状态 $\mathbf{h}_k$ 应用单步欧拉离散，得到中间状态 $\mathbf{h}'_k = \mathbf{h}_k + \text{Att}(\mathbf{h}_k)$。随后将其作为对流动力学的初始条件：

$$\frac{d\mathbf{h}(t)}{dt} = \text{FFN}(\mathbf{h}(t)) \tag{34}$$

类似地，应用第二步欧拉离散得到 $\mathbf{h}_{k+1} = \mathbf{h}'_k + \mathrm{FFN}(\mathbf{h}'_k)$。将 $\mathbf{h}'_k$ 代回即恢复标准 Transformer 的精确形式。

然而，Lie-Trotter 分裂具有非对称性（即算子顺序影响结果），且仅为一阶近似。为获得更高精度和稳定性，可采用 **Strang 分裂格式**（亦称对称分裂），该格式提供二阶近似 [Strang, 1968]。具体而言，Strang 分裂格式采用算子的对称排列方式。它并非简单先应用 $\mathrm{Att}(\cdot)$ 再应用 $\mathrm{FFN}(\cdot)$，而是先执行一个算子的半步更新，再执行另一个算子的完整步更新，最后再执行第一个算子的另半步更新。数学上，若选择分裂前馈网络算子，则 Transformer 层的更新规则对应如下复合序列：

$$\mathbf{h}_{k+1/3} = \mathbf{h}_k + \frac{1}{2} F_{\text{ffn}}(\mathbf{h}_k, \pi_k^{(1)}) \tag{35}$$

$$\mathbf{h}_{k+2/3} = \mathbf{h}_{k+1/3} + F_{\text{att}}(\mathbf{h}_{k+1/3}, \omega_k) \tag{36}$$

$$\mathbf{h}_{k+1} = \mathbf{h}_{k+2/3} + \frac{1}{2} F_{\text{ffn}}(\mathbf{h}_{k+2/3}, \pi_k^{(2)}) \tag{37}$$

这意味着一种 “三明治” 结构：一个多头自注意力子层被置于两个前馈网络子层之间。上标 (1) 和 (2) 表示，虽然数学上的 Strang 分裂通常复用相同算子，但在神经网络语境中，我们可以用不同参数 $\pi_k^{(1)}$ 和 $\pi_k^{(2)}$ 分别参数化两个前馈网络子层以增强表达能力。然而理论上，仅当参数共享时（即 $\pi_k^{(1)} = \pi_k^{(2)}$），此结构才能保证局部截断误差提升至 $O((\Delta t)^3)$。

算子分裂格式与 Transformer 架构之间的关联为架构设计提供了有力工具：我们不再仅依赖试错法，而是可以通过对底层连续动力学应用高阶数值离散与分裂技术，推导出新颖且可能更优的网络结构。例如，其他高阶分裂方法或自适应分裂方案理论上可启发 Transformer 的进一步变体。

2.2.2 基于高阶方法的架构

欧拉方法是一阶方法。一个自然的改进是考虑高阶数值求解器，它们使用更复杂的更新规则以实现更高的精度。

龙格-库塔（RK）系列是广泛使用的一类高阶求解器。欧拉方法仅使用当前状态的导数来估计解，而 RK 方法则通过聚合在多个中间点计算的导数来改进近似。例如，在 Transformer 中，标准的残差块可以被 RK 方法建模的块所替代，正如Li et al. [2022a] 所提出的。一个步长为 1 的显式 n 阶 RK 方法（其中 $n \leq 4$）的一般形式为[2]

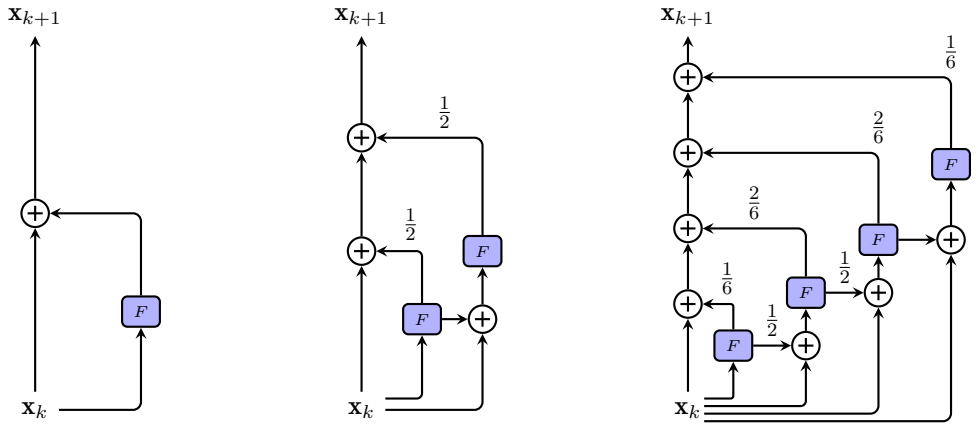
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=1}^{n} \gamma_i F_i \tag{38}$$

$$F_1 = F(\mathbf{x}_k, \theta_k) \tag{39}$$

$$F_i = F\left(\mathbf{x}_k + \sum_{j=1}^{i-1} \beta_{ij} F_j, \theta_k\right) \tag{40}$$

其中 γ_i, β_{ij} 是由特定 RK 方案（例如经典的 RK4）确定的系数。在此架构中，每个 F_i 充当斜率的中间估计，允许网络在最终更新前探测函数在多个点处的形态。理论上，此 RK 方法实现了

2. 严格来说，方程中求和上限 n 代表级数（函数求值次数），而阶指的是截断误差的收敛速率。对于显式 RK 方法，仅当阶数 ≤ 4 时，级数才等于阶数。要达到 $p > 4$ 阶，需要严格多于 p 级（例如，一个 5 阶方法至少需要 6 级）。为简化起见，本文假设 $n \leq 4$。



(a) Pre-norm (b) RK2 (c) RK4

图 5: 基于数值 ODE 求解器启发的 Transformer 子层架构对比 [Li et al., 2022a]。子图 (a)：基线预归一化架构代表了基于欧拉方法的单步更新。子图 (b) 和 (c)：RK2 和 RK4 架构在单个块内使用多次函数评估（阶段）。输出计算为这些中间状态的加权聚合，如边上的系数所示。

$O((\Delta t)^{n+1})$ 的局部截断误差。注意，参数 θ_k 在块内是共享的。虽然我们需要对函数 $F(\cdot)$ 进行 n 次求值，但每次求值都复用相同的参数 θ_k。这意味着高阶 RK 方法以增加计算成本为代价来获得更高的精度，但它是参数高效的。

例如，一个基于 RK2 的 Transformer 子层可以表述为

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{2}(F_1 + F_2) \tag{41}$$

$$F_1 = F(\mathbf{x}_k, \theta_k) \tag{42}$$

$$F_2 = F(\mathbf{x}_k + F_1, \theta_k) \tag{43}$$

在此配置中，F_1 充当预测器，提供斜率的初始猜测，而 F_2 充当“精炼器”，基于估计的未来状态调整更新。图5展示了不同阶数 RK 方法的图示。

请注意，这些基于 RK 的架构为扩展神经网络引入了一个新的维度。当前 LLM 的扩展主要侧重于增加序列长度 [Guo et al., 2025] 或层间深度 [Yang et al., 2024]，而 RK 方案则实现了层内深度扩展。这种方法提供了一种参数高效的机制，将计算预算与参数数量和序列长度解耦。

2.2.3 基于隐式方法的架构

前文所述的欧拉方法与标准龙格-库塔方法均属于**显式方法**范畴，其通过当前状态 $\mathbf{x}_{k+1}$ 直接计算下一状态 $\mathbf{x}_k$。然而显式方法可能存在数值稳定性不足的问题。为增强模型鲁棒性，可采用**隐式方法**——该方法通过包含未知未来状态自身的方程来定义下一状态。

基于隐式方法设计架构具有挑战性，因为求解 $\mathbf{x}_{k+1}$ 需要复杂的求根运算。因此通常需要采用更高级的方法，例如 PCformer [Li et al., 2024] 就采用了预测-校正范式，结合高阶显式预测器（如 RK 方法）与多步隐式校正器。

在预测阶段，PCformer 采用高阶显式求解器（如四阶 RK 方法）替代简单欧拉更新，获得高质量初始估计 $\hat{\mathbf{x}}_{k+1}$。例如可利用式 (41-43) 获得二阶 RK 估计。作为改进，PCformer 采用**指数**

移动平均（EMA）进行系数学习（即确定式 (38) 中的 γ_i）。该方法假设高阶中间近似（如 RK4 中的 F_4）比低阶近似（如 F_1）更精确，因此应对输出贡献更大权重。PCformer 摒弃标准数值方法的固定系数（如 RK4 的 $1/6, 2/6, 2/6, 1/6$），转而采用 EMA 衰减因子 b 分配权重。对于 n 阶预测器，其输出由下式给出：

$$\hat{\mathbf{x}}_{k+1} \quad = \quad \mathbf{x}_k + \sum_{i=1}^{n} b(1-b)^{n-i} F_i, \tag{44}$$

其中 b 为可学习参数（初始值设为 0.5）。

在校正阶段，模型采用隐式线性多步法（如 Adams-Moulton）。由于隐式方法需要下一时间步的函数值 $F(\mathbf{x}_{k+1})$（此时未知），故使用预测器输出 $\hat{\mathbf{x}}_{k+1}$ 作为替代。最终更新规则结合当前评估值与历史状态的加权组合：

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha F(\hat{\mathbf{x}}_{k+1}, \theta_k) + \sum_{j=0}^{m} \beta_j F(\mathbf{x}_{k-j}, \theta_{k-j}), \tag{45}$$

其中 α 与 β_j 为可学习系数，m 表示模型为校正当前状态所记忆的先前层数。该方程作为可学习的 Adams-Moulton 校正器，通过将经典固定系数替换为可学习权重 α 和 β_j，能更好地适应数据分布。这种预测-校正架构虽能减少截断误差并提升生成质量，但由于引入更多计算步骤，其推理效率可能低于标准 Transformer。

为提高推理效率，可采用迭代式**隐式欧拉 Transformer**（IIET）架构 [Liu et al., 2025]。与仅用当前状态计算下一状态的显式方法不同，隐式欧拉法定义为 $\mathbf{x}_{k+1} = \mathbf{x}_k + F(\mathbf{x}_{k+1})$。由于 $\mathbf{x}_{k+1}$ 在等式两侧同时出现，直接求解该方程具有挑战性。IIET 通过在每层内部采用定点迭代作为求解器来解决此问题。

具体而言，首先通过初始估计 $\mathbf{x}_{k+1}^{(0)}$（通常由显式欧拉步获得）进行 r 次迭代逐步优化：

$$\mathbf{x}_{k+1}^{(0)} \quad = \quad \mathbf{x}_k + F(\mathbf{x}_k, \theta_k) \tag{46}$$

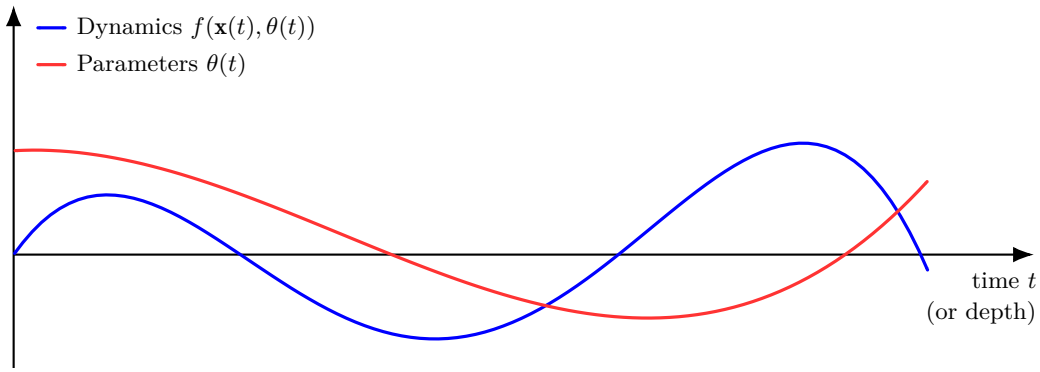
$$\mathbf{x}_{k+1}^{(i)} \quad = \quad \mathbf{x}_k + F(\mathbf{x}_{k+1}^{(i-1)}, \theta_k), \quad \text{for } i = 1, \ldots, r. \tag{47}$$

该层的最终输出为 $\mathbf{x}_{k+1} = \mathbf{x}_{k+1}^{(r)}$。这种迭代优化比单步方法更能逼近隐式方程的真实解。为进一步降低推理成本，可根据各层对最终输出的贡献度，对迭代次数 r 实施动态剪枝。

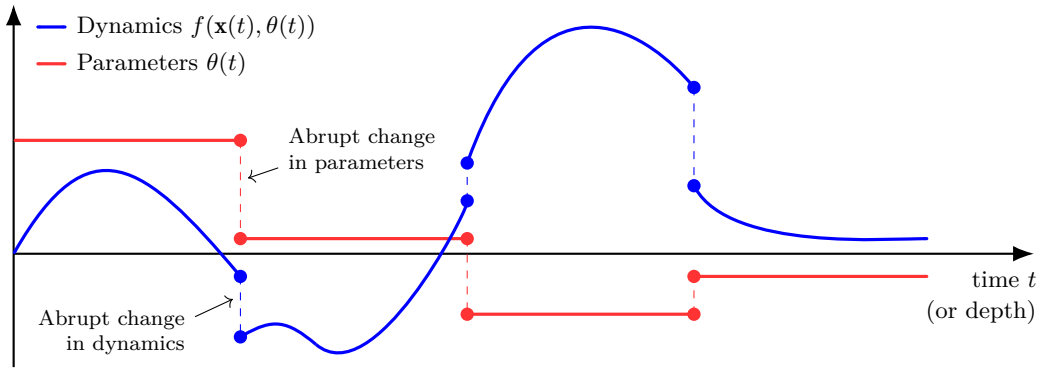
2.3 关于稳定性与刚性的评注

尽管常微分方程为模型设计提供了理论解释，但在将相关概念应用于深度神经网络时，底层动力系统的稳定性仍然是一个挑战。从常微分方程的角度来看，Transformer 的稳定性与信息流的平滑性以及常微分方程的刚性密切相关。

将 Transformer 建模为常微分方程的一个问题是，学习到的动力学可能是刚性的。这里的**刚性**指的是隐藏状态连续动力学以极不相同的速率演化的现象，例如，某些分量变化迅速，而其他分量变化缓慢。因此，刚性常微分方程是指标准显式数值方法严重不稳定的常微分方程，需要非常小的步长才能稳定地求解该方程。在神经网络的背景下，这意味着标准显式求解器（或固定深度的残差连接）可能无法有效地传播信号，因为在尝试捕捉这些快速的瞬态变化时，若没有足够精细的离散化，它们容易产生数值不稳定性。正如经典数值分析文献所指出的 [Hairer and Wanner,



(a) Smooth evolution of system dynamics and parameters over time



(b) Sharp transitions in system dynamics caused by abrupt changes in parameters

图 6: 系统动力学因参数突变而发生变化的示意图。蓝线表示系统动力学 $f(\mathbf{x}(t), \theta(t))$ 的演化，红线表示参数 $\theta(t)$ 的演化。子图 (a) 展示了一种理想情况，其中参数和动力学均随时间平滑且连续地演化。子图 (b) 展示了一种现实情况，其中离散层中的参数表现为阶跃函数。在特定时间步参数发生的突变导致了系统动力学的急剧转变。

1996]，标准欧拉法或龙格-库塔法等显式方法的稳定区域有限。当应用于刚性问题时，它们通常需要很小的步长以避免数值发散。相比之下，隐式方法（例如，2.2.3 节中介绍的方法）通常更适用于刚性问题，因为它们具有更大的稳定区域。

对于 Transformer 而言，由于层的离散性，这个问题更为严重。在这些模型中，参数 θ 是独立的，并且在不同层之间可能存在巨大差异。这与稳定求解常微分方程所需的平滑动力学假设相矛盾。回顾常微分方程公式 $\frac{d\mathbf{x}(t)}{dt} = F(\mathbf{x}(t), \theta(t))$，其中 $\theta(t)$ 表示随时间 t 连续变化的参数。然而，在标准 Transformer 中，每层的参数是固定的，从而将 $\theta(t)$ 离散化为一个阶梯函数（即 $\theta_1, \theta_2, \dots$）。这些参数的突变导致系统动力学从一层到下一层发生急剧变化。例如，在大语言模型中，经常观察到前几层的参数值明显不同，导致不同层之间的中间表示存在显著差异。图 6 展示了此问题的示意图。

为了缓解由刚性和参数突变引起的数值不稳定性，很自然地会约束函数 F 的变异性。实现此目标的一种常见方法是通过 Lipschitz 连续性条件。具体来说，为确保常微分方程良态且能稳定求解，必须存在一个常数 K 使得 $\|F(\mathbf{x}_1) - F(\mathbf{x}_2)\| \leq K\|\mathbf{x}_1 - \mathbf{x}_2\|$。在神经网络中，Lipschitz 常数与每层权重矩阵的谱范数密切相关 [Neyshabur, 2017; Miyato et al., 2018]。谱范数定义为矩阵的最大奇异值。Gouk et al. [2021] 证明，约束谱范数可以有效地限制 Lipschitz 常数。这种约束不仅平滑了动力学，还确保了信号通过网络深度的放大受到控制。这有助于防止数值发散。

更广泛地，我们可以使用显式正则化技术来实现更高的稳定性。可以考虑两种常见策略：

- **参数正则化**。为了强制 Transformer 中动力学随深度的演化更平滑，可以在损失函数中引入一个惩罚项。以下方程展示了一个惩罚项的例子 [Haber and Ruthotto, 2017]

$$\mathcal{L}_{\text{param}} = \lambda_{\text{param}} \sum_{k=1}^{L-1} ||\theta_{k+1} - \theta_k||_2^2, \tag{48}$$

  其中 λ_{param} 是惩罚项的权重，L 是 Transformer 层堆叠的深度[3]。此项约束相邻层具有相似的参数。在极限情况下（例如，当 $\lambda_{\text{param}} \to \infty$ 时），此约束强制 $\theta_{k+1} \approx \theta_k$，这导致了参数共享架构 [Dehghani et al., 2019; Lan et al., 2020]。在此类模型中，θ 在各层间变为常数，非自治常微分方程转化为一个自治系统（时不变）。这实际上将深度维度视为循环神经网络的时间步长。一般而言，自治系统更稳定且参数效率更高。

- **状态正则化**。此方法直接控制层间输出的变化。例如，我们可以惩罚相邻层输出之间的较大差异，如下所示

$$\mathcal{L}_{\text{state}} = \lambda_{\text{state}} \sum_{k=1}^{L-1} ||\mathbf{x}_{k+1} - \mathbf{x}_k||_2^2, \tag{49}$$

  其中 λ_{state} 是该正则化项的权重。从这个意义上说，像层归一化这样的归一化方法可以被视为在 Transformer 中正则化隐藏状态的一种有效方式。另一种方法涉及将先前层的输出作为输入来产生组合输出。此类层组合技术已广泛用于训练深度神经网络 [Huang et al., 2017; Wang et al., 2018]。从数值分析的角度来看，这些架构可以解释为多步数值积分器（例如，线性多步方法），其中下一状态的近似依赖于先前状态的历史，而不仅仅是紧邻的前一个状态。这种序列依赖有效地平滑了隐藏状态 $\mathbf{x}(t)$ 的轨迹。通过聚合来自多个先前步骤的信息，即使离散变换变化迅速，信号的传播也能相对稳健。

3. 神经常微分方程与流模型

在上一节中，我们将离散的残差网络和 Transformer 解释为常微分方程（ODE）的离散化近似。此类模型的一个显著局限在于，其离散化过程使用了固定的步长（即 $\Delta t = 1$），这阻碍了模型根据输入实例的复杂度自适应地调整其计算量。这种粗粒度的离散化通常不适合需要细粒度建模的问题，因为在这些问题中，底层动力学可能快速演化，需要更小的时间步长才能进行精确近似。此外，由于步数是在模型设计阶段确定的，网络深度变成了一个静态的超参数，而非自适应的属性。

3. 此处，一个 Transformer 层包含两个子层。θ_k 表示整个层的所有参数集合。

在本节中，我们首先探讨 **神经常微分方程** [Chen et al., 2018]，它通过结合神经网络和连续时间常微分方程来模拟系统动力学的演化。神经常微分方程的一个重要特性是，它将网络深度视为一个连续的时间变量。模型不再指定一个固定的离散层序列，而是通过一个连续时间常微分方程来定义隐藏状态的导数，最终输出通过使用黑盒 ODE 求解器求解该常微分方程获得。此外，我们将描述神经常微分方程的训练方法，特别是伴随灵敏度方法，该方法为优化此类模型提供了一种内存高效的方式。最后，我们讨论 **连续归一化流**的概念，这是神经常微分方程在描述概率密度连续演化方面的自然扩展。该框架为后续关于生成建模的讨论奠定了基础（参见第 4 节）。

3.1 神经常微分方程

我们已经看到，在残差网络中，隐藏状态的更新遵循 $\mathbf{x}_{l+1} = \mathbf{x}_l + \Delta t \cdot F(\mathbf{x}_l, \theta_l)$ 这一公式。当 $\Delta t \to 0$ 时，该方程对应于连续时间常微分方程的欧拉离散化：

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta(t)). \tag{50}$$

此处我们使用两种函数符号 F 和 f 来分别区分离散残差映射与状态的瞬时变化率。

两种模型的关键区别在于参数化方式。标准残差网络中，每个层 l 都有其独立的参数集 θ_l。而在式 (50) 表达的连续极限中，参数 $\theta(t)$ 可以随时间连续变化。不过实践中，标准神经常微分方程通常在整个积分区间使用共享参数集 θ。为了使动力学行为能随深度（时间）变化，当前时间 t 会被显式地作为网络输入。因此动力学可表述为非自治常微分方程：

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta, t). \tag{51}$$

用连续时间变量 t 替代层索引 l 的设计，使得模型能够在任意时间点而非固定时间步长上评估状态。

这类模型被称为*神经*常微分方程，是因为动力学函数 $f(\cdot)$ 由神经网络实现。换言之，我们使用神经网络来估计系统的动力学行为。给定一个神经常微分方程，与监督学习问题类似，存在两个基本过程：

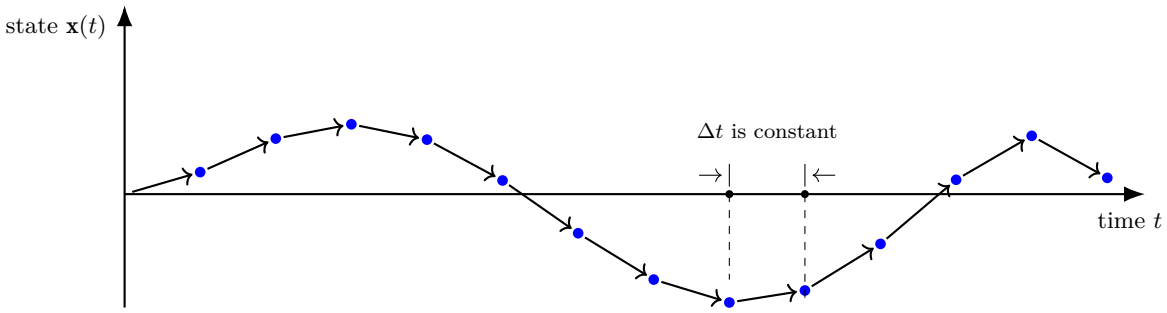
- **推断**（前向传播）。推断过程旨在给定参数的情况下计算任意时刻的状态。
- **训练**（反向传播）。训练过程旨在优化参数 θ 以最小化损失函数。

在推理阶段，模型的输入是初始状态 $\mathbf{x}(t_0)$，输出是最终时刻 t_1 的状态 $\mathbf{x}(t_1)$。也就是说，前向传播本质上就是求解这个初值问题的过程。输出状态 $\mathbf{x}(t_1)$ 作为常微分方程的解，通过积分动力学方程获得：

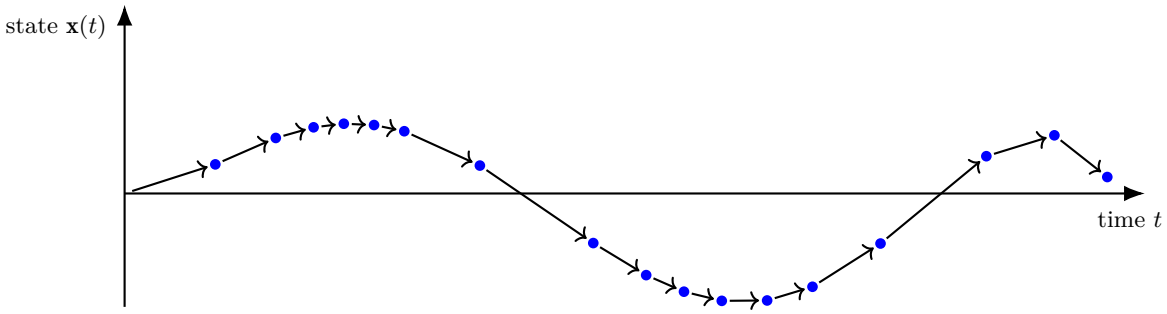
$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + \int_{t_0}^{t_1} f(\mathbf{x}(\tau), \tau; \theta) d\tau. \tag{52}$$

由于复杂神经网络 f 的积分无法解析求解，输出需要通过数值常微分方程求解器（如欧拉法、龙格-库塔法或更先进的方法）计算得到。这使得下式可被求值：

$$\mathbf{x}(t_1) = \text{ODESolve}(\mathbf{x}(t_0), f, t_0, t_1, \theta). \tag{53}$$



(a) A system whose state evolves with a fixed time step (like residual networks)



(b) A system whose state evolves continuously or with a dynamic time step (like neural ODEs)

图 7: 固定步长与动态步长求解常微分方程的图示。子图 (a) 展示了使用恒定步长生成的轨迹，类似于残差网络中的离散步骤（可视为常微分方程的欧拉离散化）。子图 (b) 展示了使用动态步长求解器获得的轨迹，该求解器可用于模拟连续状态演化的神经常微分方程。

其中 ODESolve$(\cdot)$ 表示通用常微分方程求解器，通常采用现成的数值积分工具实现。这种表述将动力学定义与轨迹计算解耦，因此求解器能根据轨迹复杂度自适应调整步长——这是固定深度残差网络所不具备的重要特性。图 7 展示了固定步长与动态步长求解常微分方程的对比示意图。

神经常微分方程的训练可视为标准深度神经网络训练任务，通过计算损失函数对参数 θ 的梯度来迭代更新参数。直接方法是沿求解器运算过程执行标准反向传播，但这往往因需要存储整个轨迹的中间状态以应用链式法则而产生高昂内存开销。内存瓶颈可能限制精细求解器或长时积分的使用。为此，神经常微分方程通常采用**伴随灵敏度方法**进行训练，该方法能以与积分步数无关的恒定内存成本计算梯度。我们将在第 3.2 节详细讨论训练细节。

简而言之，神经常微分方程使用神经网络 $f(\cdot)$ 来近似系统动力学 $\frac{d\mathbf{x}(t)}{dt}$。在训练阶段，我们优化该网络以建模这些动力学行为；在测试阶段，则可通过求解神经常微分方程直接恢复系统在任意时刻的状态。

3.2 伴随灵敏度方法

伴随灵敏度方法（或称伴随方法）是源自最优控制领域的一种经典技术，用于计算由微分方程支配的系统的灵敏度。在深度学习中，它可以被解释为连续时间的反向传播。其核心思想是引入一组辅助变量，称为**伴随状态**，用于追踪损失的梯度如何随时间反向演化。它通过求解第二个微分方程来计算模型参数的梯度，而不是将链式法则显式地应用于前向传播的离散操作。

在将其与标准反向传播进行比较时，应强调这一区别。在一个朴素的实现（通常称为“先离散化再优化”）中，人们会将数值 ODE 求解器视为具有有限层数的计算图，并直接通过求解器的内部操作进行反向传播。然而，由于 ODE 求解器可能需要数千个小步骤才能达到高精度，标准反向传播需要存储每个步骤的中间激活值。这导致内存使用量随积分步数线性增长（$O(T)$），达到难以承受的程度。换句话说，使用大量步长进行 ODE 求解的离散化使得训练内存密集，有时甚至不可行。

形式上，我们将伴随状态 $\mathbf{a}(t)$ 定义为损失函数 $\mathcal{L}$ 相对于任意给定时间 t 的隐藏状态 $\mathbf{x}(t)$ 的梯度

$$\mathbf{a}(t) \quad = \quad \frac{\partial \mathcal{L}}{\partial \mathbf{x}(t)}. \tag{54}$$

直观上，$\mathbf{a}(t)$ 表示最终损失对时间 t 时状态微小扰动的敏感性。由于损失是基于最终状态 $\mathbf{x}(t_1)$ 计算的，因此伴随状态的边界条件是已知的

$$\mathbf{a}(t_1) \quad = \quad \frac{\partial \mathcal{L}}{\partial \mathbf{x}(t_1)}. \tag{55}$$

需要注意的是，虽然 $\mathbf{x}(t)$ 是从 t_0 到 t_1 正向求解的，但伴随状态 $\mathbf{a}(t)$ 必须使用 $\mathbf{a}(t_1)$ 作为反向积分的初始值，从 t_1 到 t_0 反向求解。图 8 展示了该方法的高层示意图。

3.2.1 伴随动力学

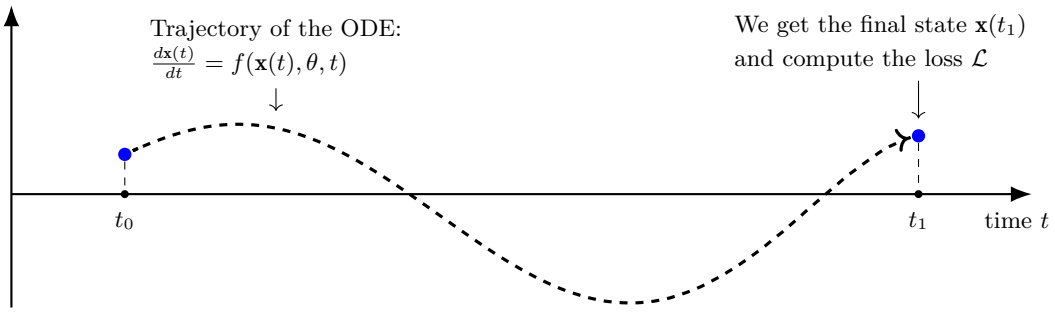
我们不通过求解器的步骤进行反向传播，而是通过求解一个新的常微分方程来计算 $\mathbf{a}(t)$。为了理解 $\mathbf{a}(t)$ 的动力学，考虑使用步长为 Δt 的简单欧拉方法对前向传播进行离散化。从 t 到 $t+\Delta t$ 的状态更新为

$$\mathbf{x}(t+\Delta t) \quad = \quad \mathbf{x}(t) + \Delta t \cdot f(\mathbf{x}(t), \theta, t). \tag{56}$$

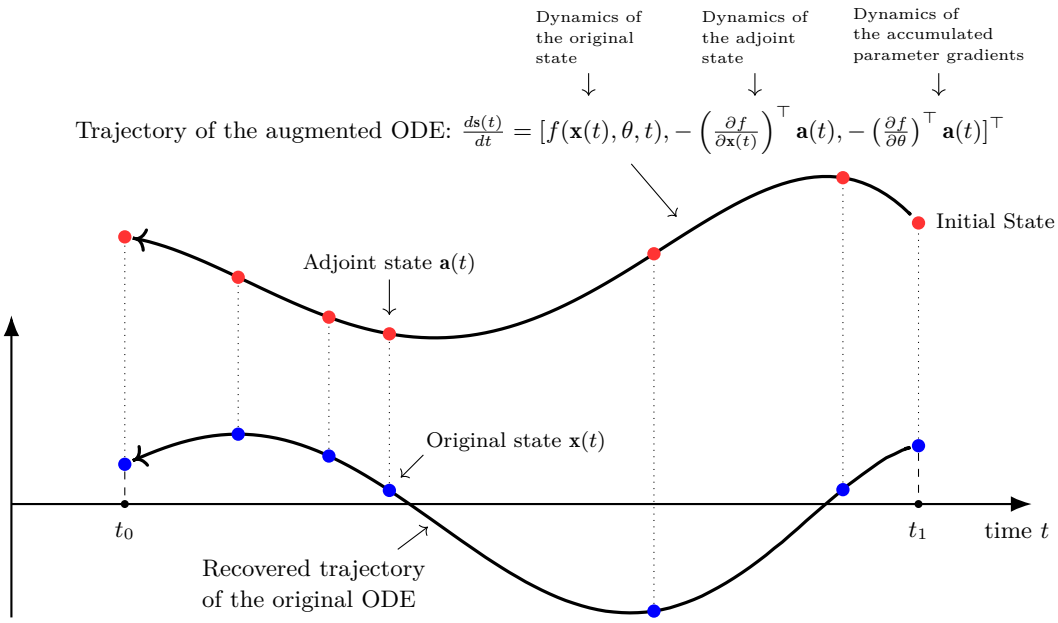
应用链式法则，损失关于隐藏状态 $\mathbf{x}(t)$ 的梯度可以表示为

$$\begin{aligned} \mathbf{a}(t) &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}(t)} \\ &= \left(\frac{\partial \mathbf{x}(t+\Delta t)}{\partial \mathbf{x}(t)}\right)^\top \frac{\partial \mathcal{L}}{\partial \mathbf{x}(t+\Delta t)} \\ &= \left(\frac{\partial \mathbf{x}(t+\Delta t)}{\partial \mathbf{x}(t)}\right)^\top \mathbf{a}(t+\Delta t), \end{aligned} \tag{57}$$

其中 $\frac{\partial \mathbf{x}(t+\Delta t)}{\partial \mathbf{x}(t)} \in \mathbb{R}^{d\times d}$ 是雅可比矩阵，而 $\mathbf{a}(t), \mathbf{a}(t+\Delta t) \in \mathbb{R}^d$ 是列向量。



(a) Forward pass (integrating the dynamics of the ODE from t_0 to t_1)



(b) Backward pass (integrating the dynamics of the augmented ODE from t_1 to t_0)

图 8: 神经常微分方程中伴随敏感性方法的示意图 [Chen et al., 2018]。该方法包含两个步骤。在前向传播过程中（子图 (a)），状态 $\mathbf{x}(t)$ 通过积分由神经网络 $f(\mathbf{x}(t), \theta, t)$ 定义的动力学，从 t_0 演化到 t_1。最终状态 $\mathbf{x}(t_1)$ 用于计算标量损失 $\mathcal{L}$。在反向传播过程中（子图 (b)），无需通过求解器的离散步骤进行反向传播，而是求解一个从 t_1 到 t_0 反向积分的增广常微分方程组。该增广系统同时重构原始状态轨迹 $\mathbf{x}(t)$，计算伴随状态 $\mathbf{a}(t)$（损失关于状态的梯度），并累积关于参数 θ 的梯度。这种方法使得梯度计算具有 $O(1)$ 的内存开销，因为它避免了在积分过程中存储中间状态。

将方程 (56) 代入方程 (57) 得到

$$\begin{aligned}
\mathbf{a}(t) &= \left(\frac{\partial\big(\mathbf{x}(t) + \Delta t \cdot f(\mathbf{x}(t), \theta, t)\big)}{\partial \mathbf{x}(t)}\right)^\top \mathbf{a}(t + \Delta t) \\
&= \left(\frac{\partial \mathbf{x}(t) + \partial \Delta t \cdot f(\mathbf{x}(t), \theta, t)}{\partial \mathbf{x}(t)}\right)^\top \mathbf{a}(t + \Delta t) \\
&= \left(\mathbf{I} + \Delta t \cdot \frac{\partial f(\mathbf{x}(t), \theta, t)}{\partial \mathbf{x}(t)}\right)^\top \mathbf{a}(t + \Delta t). \qquad (58)
\end{aligned}$$

重新整理此方程，得到伴随状态的差商

$$\frac{\mathbf{a}(t+\Delta t)-\mathbf{a}(t)}{\Delta t} = -\left(\frac{\partial f(\mathbf{x}(t),\theta,t)}{\partial \mathbf{x}(t)}\right)^{\top}\mathbf{a}(t+\Delta t). \tag{59}$$

通过取极限 $\Delta t \to 0$，我们得到伴随状态的连续时间动力学。这导出了描述 $\mathbf{a}(t)$ 瞬时变化率的微分方程

$$\frac{d\mathbf{a}(t)}{dt} = -\left(\frac{\partial f(\mathbf{x}(t),\theta,t)}{\partial \mathbf{x}(t)}\right)^{\top}\mathbf{a}(t). \tag{60}$$

这里，$\frac{\partial f}{\partial \mathbf{x}}$ 是动力学函数的雅可比矩阵。这个方程告诉我们，伴随状态由一个线性常微分方程定义，该方程依赖于原始函数 $f(\cdot)$ 的雅可比矩阵。

类似地，我们可以推导出参数 θ 的梯度。在每个时间步 t，参数 θ 对状态变化做出微小贡献，进而影响损失。时间 t 处的瞬时梯度贡献是损失对状态的敏感度 ($\mathbf{a}(t)$) 乘以状态变化对参数的敏感度 ($\frac{\partial f}{\partial \theta}$)。总梯度是这些瞬时贡献在整个轨迹上的累积（积分）

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\int_{t_1}^{t_0}\left(\frac{\partial f(\mathbf{x}(t),\theta,t)}{\partial \theta}\right)^{\top}\mathbf{a}(t)dt. \tag{61}$$

注意，积分是从 t_1 到 t_0 反向进行的，这与反向传播的反向过程一致。

3.2.2 增广常微分方程与内存效率

式 (60) 和 (61) 给出了精确的梯度，但它们依赖于所有时刻 t 的状态轨迹 $\mathbf{x}(t)$。在标准的反向传播中，这需要存储前向传播中的所有中间状态 $\mathbf{x}(t)$，导致 $O(T)$ 的内存开销。

伴随灵敏度方法的关键优势在于通过反向时间重建状态轨迹来避免这种存储。由于描述状态演化的常微分方程是确定性的且通常是可逆的，我们可以通过反向运行动力学从最终状态 $\mathbf{x}(t_1)$ 恢复 $\mathbf{x}(t)$[4]。

为了高效地计算梯度，我们构建一个包含原始状态、伴随状态和累积参数梯度的**增广状态**

$$\mathbf{s}(t) = \begin{bmatrix}\mathbf{x}(t)\\ \mathbf{a}(t)\\ \frac{\partial \mathcal{L}}{\partial \theta}(t)\end{bmatrix}, \tag{62}$$

其中 $\frac{\partial \mathcal{L}}{\partial \theta}(t)$ 是从 t_1 到 t 的反向积分，即 $\frac{\partial \mathcal{L}}{\partial \theta}(t) = -\int_{t_1}^{t}\left(\frac{\partial f(\mathbf{x}(\tau),\theta,\tau)}{\partial \theta}\right)^{\top}\mathbf{a}(\tau)d\tau$。这个组合系统的动力学从 t_1 到 t_0 联合反向求解：

$$\frac{d\mathbf{s}(t)}{dt} = \begin{bmatrix} f(\mathbf{x}(t),\theta,t)\\ -\left(\frac{\partial f}{\partial \mathbf{x}(t)}\right)^{\top}\mathbf{a}(t)\\ -\left(\frac{\partial f}{\partial \theta}\right)^{\top}\mathbf{a}(t)\end{bmatrix}, \tag{63}$$

4. 常微分方程在简单的函数意义上并非天生可逆，但这一概念适用于机器学习中的神经常微分方程和流（即随时间变化的解）。非正式地说，可逆性意味着逆转变换以找到先前的状态或在空间之间映射。

初始条件为

$$\mathbf{s}(t_1) \quad = \quad \begin{bmatrix} \mathbf{x}(t_1) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{x}(t_1)} \\ \mathbf{0} \end{bmatrix}. \tag{64}$$

增广动力学的第一个分量就是原始的常微分方程。这意味着当求解器反向积分时，它会即时重建轨迹 $\mathbf{x}(t)$。在任何特定时间 t，评估雅可比矩阵（在第二和第三分量中）所需的 $\mathbf{x}(t)$ 值在当前增广状态 $\mathbf{s}(t)$ 中即可获得。因此，我们不需要存储前向传播的中间状态，内存复杂度从 $O(T)$ 降低到 $O(1)$。本质上，伴随方法以重新求解状态轨迹的计算开销为代价，换取了内存的节省。需要注意的是，这个增广常微分方程可以使用现成的数值求解器来求解。人们也可以根据特定需求，通过简单地调整容差参数来平衡计算效率和数值精度。

总而言之，神经常微分方程的训练过程可以概括为以下两个步骤：

1. **前向传播：** 从 t_0 到 t_1 求解常微分方程以得到 $\mathbf{x}(t_1)$，并计算损失 $\mathcal{L}$。
2. **反向传播：** 在 t_1 处构造增广状态的初始值：$\mathbf{s}(t_1) = [\mathbf{x}(t_1), \frac{\partial \mathcal{L}}{\partial \mathbf{x}(t_1)}, \mathbf{0}]^\top$。使用常微分方程求解器从 t_1 到 t_0 反向积分增广动力学方程 $\frac{d\mathbf{s}(t)}{dt} = [f(\mathbf{x}(t), \theta, t), -\left(\frac{\partial f}{\partial \mathbf{x}(t)}\right)^\top \mathbf{a}(t), -\left(\frac{\partial f}{\partial \theta}\right)^\top \mathbf{a}(t)]^\top$。

3.3 神经动力学与流

神经常微分方程为描述系统动力学提供了强大的工具。一个关键洞见是，我们可以使用神经网络 $f(\cdot)$ 来建模一个向量场。这定义了一个流，它表示为一个连续、可逆的变换，该变换使状态空间随时间演化。在本小节中，我们将形式化这些概念，从单个状态轨迹转向整个概率分布的演化。这个框架将作为后续章节中提出的生成模型的基础。

3.3.1 向量场

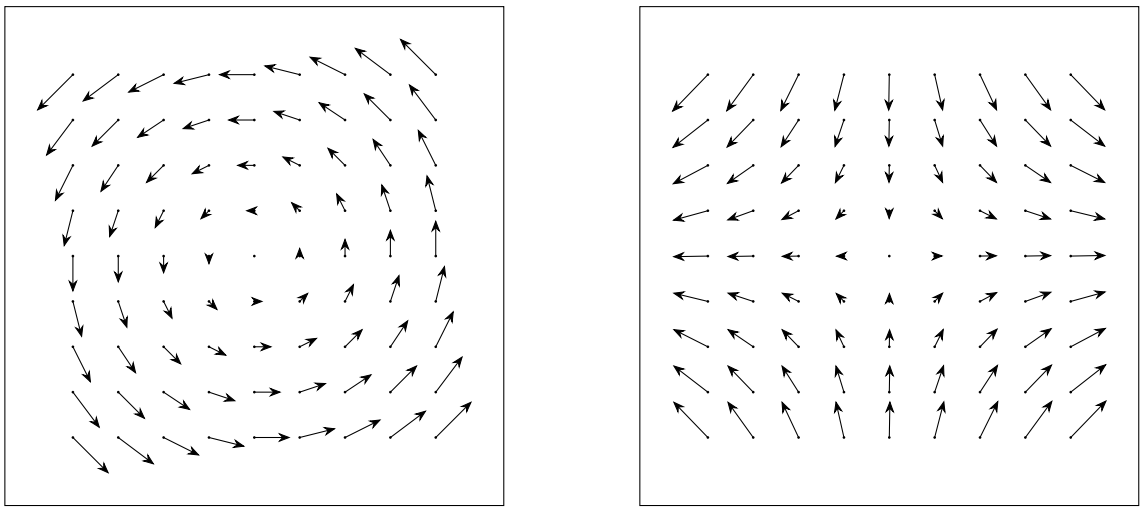
向量场是一种数学概念，用于描述空间中每个点都关联有特定大小和方向的场。为了直观地理解，想象一条流动的河流表面。在任何给定的位置和特定的时刻，水都以特定的速度和方向运动。如果我们在河流的每个点都画一个箭头来表示该处的局部速度，那么这些箭头的集合就构成了一个向量场。另一个与我们之前讨论更接近的例子是，想象状态空间充满了一种流体。在这个空间中的每一点，流体都以特定的速度和方向流动。这种将速度矢量分配给空间和时间中每一点的方式，就称为向量场。

形式上，$\mathbb{R}^d$ 上的一个向量场是一个映射

$$f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d. \tag{65}$$

它为每个点 $\mathbf{x}$ 在时间 t 分配一个速度矢量。在神经常微分方程的背景下，我们使用神经网络 $f(\mathbf{x}(t), \theta, t)$ 对这个场进行参数化。给定当前状态 $\mathbf{x}(t)$ 和时间 t，网络通过常微分方程 $\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta, t)$ 输出时间导数，这指示了“数据当前移动得有多快以及朝哪个方向移动”。请注意，这里的向量场是时间依赖的，因为一个点处的速度会随时间变化。

图 9 展示了平面上的两个示例向量场。第一个是由 $f(\mathbf{x}) = [-x_2, x_1]^\top$ 定义的向量场。放置在该场中的粒子将垂直于其位置矢量运动，从而描绘出一个圆形轨迹。第二个是由 $f(\mathbf{x}) = [\sin(x_1) +$



(a) Vector field $f(\mathbf{x}) = \begin{bmatrix} -x_2 \\ x_1 \end{bmatrix}$

(b) Vector field $f(\mathbf{x}) = \begin{bmatrix} \sin(x_1) + \sin(x_2) \\ \sin(x_1) - \sin(x_2) \end{bmatrix}$

图 9: 二维平面中向量场的两个示例。每个箭头表示该位置处的瞬时速度向量。子图 (a) 展示了一个线性系统，呈现围绕原点的圆形流线。子图 (b) 展示了一个由正弦函数定义的非线性系统，形成了固定点与流动模式的周期性排列。

$\sin(x_2), \sin(x_1) - \sin(x_2)]^\top$ 定义的向量场。这产生了更复杂的流动模式。在实际应用中，通过使用深度神经网络对 $f(\cdot)$ 进行参数化，我们可以建模高度复杂、非线性且时变的动力学。

3.3.2 流与微分同胚

向量场描述了任意给定点的瞬时速度，而流则描述了点在该空间中运动的长期轨迹。回到河流的类比，向量场告诉我们当前每个具体坐标处水的流速。而流则告诉我们，一片在特定位置落入河流的叶子，在一段时间后最终会到达何处。

数学上，流映射 Φ 是一个函数，用于追踪所有可能初始状态随时间的演化。形式上，我们将流定义为一个映射 $\Phi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$，它接受一个初始点 $\mathbf{x}(0)$ 和一个时间参数 t，并产生在时间 t 的状态。这是以下初值问题的解

$$\frac{\partial \Phi(\mathbf{x}(0), t)}{\partial t} \quad = \quad f(\Phi(\mathbf{x}(0), t), \theta, t), \tag{66}$$

并满足初始条件

$$\Phi(\mathbf{x}(0), 0) \quad = \quad \mathbf{x}(0). \tag{67}$$

为了分析整个空间的演化，我们通常固定时间 t，并考虑时间-t 映射，记作 $\Phi_t : \mathbb{R}^d \to \mathbb{R}^d$，其中 $\Phi_t(\cdot) = \Phi(\cdot, t)$。向量场 $f(\cdot)$ 表示速度，而映射 Φ_t 则表示整个空间的最终变换。随着 t 的增加，Φ_t 将 $\mathbb{R}^d$ 中的所有点沿着由向量场 $f(\cdot)$ 定义的轨迹推动。需要注意的是，状态 $\mathbf{x}(t)$ 是流在特定

Symbol	名称	定义	类比	解释
f	向量场	导数 $\frac{d\mathbf{x}(t)}{dt}$	某一特定位置处水流的瞬时速度和时间。	神经网络本身。它定义了数据如何演化。
$\mathbf{x}(t)$	状态/轨迹	在 $\mathbb{R}^d$ 中的取值时间 t。	某一特定时刻 t 时树叶的坐标。	在某一时刻的隐藏状态。
Φ	流映射	$\Phi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$	一个通用函数用于在任意起点和任意时间下找到终点。	常微分方程的一般解，即网络随时间产生的累积效应。
Φ_t	时间-t 映射	$\Phi_t(\cdot) = \Phi(\cdot, t)$ $(\mathbb{R}^d \to \mathbb{R}^d)$	整条河流的一个快照：整个水面在 t 秒后如何发生位移。	从输入到输出的总体变换（微分同胚）。

表 2: 与流相关的概念。

初始点处的求值结果，即 $\mathbf{x}(t) = \Phi_t(\mathbf{x}(0))$。为了进一步阐明这些符号之间的区别和联系，我们在表 2 中进行了总结。

重要的是，为了使流在生成建模中定义良好且有用，它必须是一个**同胚**，并且更严格地说，是一个**微分同胚**。同胚是一个双射（一一对应且满射）映射 $g : \mathcal{X} \to \mathcal{Y}$，使得 $g(\cdot)$ 及其逆映射 $g^{-1}(\cdot)$ 都是连续的。直观地说，同胚是一种橡皮膜变形。想象状态空间是一张有弹性的橡皮膜。你可以任意拉伸、压缩或扭曲它，但不能撕裂它（这会破坏连续性），也不能将不同部分粘合在一起（这会破坏双射性）。如果一个空间可以通过同胚变换为另一个空间，则认为这两个空间在拓扑上是等价的。从神经 ODE 的角度看，流 $\Phi_t(\cdot)$ 充当了一个同胚。这确保了数据的全局结构得以保留：在初始分布中彼此接近的点，在变换后的分布中仍将保持相对接近，并且空间中不会产生空洞。

在神经 ODE 和基于流的模型中，我们通常需要一个比同胚更强的条件。我们需要变换是光滑的，以便能够进行微积分运算（例如，计算梯度和使用变量替换公式）。这引出了微分同胚的概念。

一个映射 $\Phi_t : \mathbb{R}^d \to \mathbb{R}^d$ 是微分同胚，如果：

- $\Phi_t(\cdot)$ 是一个同胚映射（它是双射的，且 $\Phi_t(\cdot)$ 和 $\Phi_t^{-1}(\cdot)$ 均存在且连续）。
- $\Phi_t(\cdot)$ 和 $\Phi_t^{-1}(\cdot)$ 都是连续可微的（C^1 或更高阶）。

流的微分同胚性质是许多基于流模型的基础。在神经 ODE 中，向量场 $f(\cdot)$ 通常由具有光滑激活函数（例如 Tanh）的神经网络参数化，使得 $f(\cdot)$ 连续可微（C^1）。根据皮卡-林德勒夫定理，虽然利普希茨连续性足以保证轨迹的唯一性（确保 Φ_t 是同胚），但网络的 C^1 光滑性进一步确保了流 $\Phi_t(\cdot)$ 是一个微分同胚[5]。

5. 此性质也称为对初始条件的光滑依赖性，这是皮卡-林德勒夫定理在动力系统和流形上的自然延伸 [Coddington, 1955]。

这一性质在许多应用中具有优势，尤其是在生成模型中。首先，因为 $\Phi_t(\cdot)$ 是微分同胚，所以变换是完全可逆的。如果我们知道系统在时间 t_1 的状态，我们可以对同一个 ODE 进行反向时间积分，以精确恢复在 t_0 的初始状态。这是精确密度估计和内存高效训练的基础。其次，基于流的变换不能改变输入空间的基本拓扑结构。例如，一个流不能将单个连通分量变换为两个分离的“岛屿”，也不能在一个原本没有打结的绳子上打结。这种路径保持的特性确保了神经网络学习的是概率质量的连续变形，而不是点的混沌重排。关于流的更多细节，可以在相关论文中找到，例如 [Lipman et al., 2024]。

3.3.3 连续归一化流

虽然标准流追踪的是单个点的运动，但连续归一化流（CNF）将此框架扩展到了整个概率分布的演化。

想象一下，我们不是释放一片叶子，而是将一团彩色染料释放到河流中。随着水流流动，染料云会变形、拉伸和压缩，以跟随复杂的水流。虽然云团的形状发生变化，但染料的总质量保持不变。CNF 定义了一个简单分布（如高斯分布）与复杂数据分布之间的可逆映射。在此语境中，术语 *归一化*指的是将复杂数据简化（或归一化）回基础分布的过程。

为了形式化概率分布的演化，我们必须定义在状态空间中对点进行操作的流 $\Phi_t(\cdot)$ 如何诱导出概率测度上的变换。令 $p_t(\mathbf{x})$ 表示时刻 t 状态的概率密度函数。该密度的演化由概率质量守恒定律支配：当空间在流的作用下膨胀或收缩时，局部密度必须反向调整以保持总质量不变。

考虑初始时刻 $t = 0$ 时初始空间中的任意区域 $\mathcal{S}_0$。在流 $\Phi_t(\cdot)$ 的作用下，该区域在时刻 t 演化为新区域 $\mathcal{S}_t = \Phi_t(\mathcal{S}_0)$。由于概率质量既不会创生也不会消灭，区域内包含的总概率必须保持不变

$$\int_{\mathcal{S}_t} p_t(\mathbf{x}) d\mathbf{x} \quad = \quad \int_{\mathcal{S}_0} p_0(\mathbf{z}) d\mathbf{z}, \tag{68}$$

其中 $\mathbf{x}$ 和 $\mathbf{z}$ 分别表示时刻 t 和 0 空间中的样本。为了关联这些积分，我们对左侧应用多元变量替换定理。我们使用映射 $\mathbf{x} = \Phi_t(\mathbf{z})$ 将积分变量 $\mathbf{x}$ 替换为其原像 $\mathbf{z}$。此替换引入了雅可比行列式的绝对值以考虑体积元的变化

$$\int_{\mathcal{S}_0} p_t(\Phi_t(\mathbf{z})) \left| \det \frac{\partial \Phi_t(\mathbf{z})}{\partial \mathbf{z}} \right| d\mathbf{z} \quad = \quad \int_{\mathcal{S}_0} p_0(\mathbf{z}) d\mathbf{z}. \tag{69}$$

从几何上看，雅可比行列式项 $\left| \det \frac{\partial \Phi_t(\mathbf{z})}{\partial \mathbf{z}} \right|$ 可视为体积膨胀因子[6]。它量化了目标空间中无穷小体积与源空间中体积的比率。该项的存在确保了积分测度被正确缩放，以反映由流引起的坐标系的扭曲。

由于式 (69) 对任意区域 $\mathcal{S}_0$ 都成立，被积函数必须逐点相等。通过设 $\mathbf{z} = \mathbf{x}(0)$ 和 $\Phi_t(\mathbf{z}) = \mathbf{x}(t)$，我们得到密度之间的显式关系

$$p_t(\mathbf{x}(t)) \left| \det \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(0)} \right| \quad = \quad p_0(\mathbf{x}(0)). \tag{70}$$

6. 例如，如果行列式为 2，则 $\mathbf{z}$ 处的无穷小体积在 $\mathbf{x}$ 处被拉伸为原来的两倍。

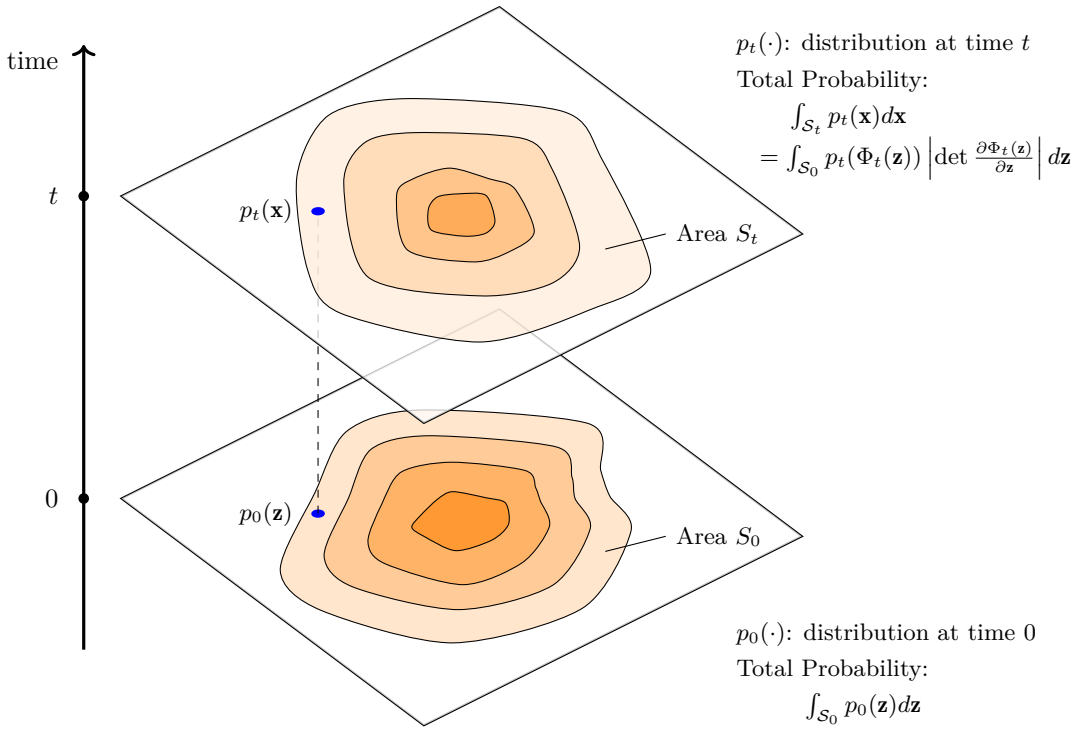


图 10: 连续归一化流中概率密度变换示意图。该图展示了概率分布随时间的连续演化过程。基础分布中的一个点 $\mathbf{z}$ 被映射为时间 t 处的 $\mathbf{x} = \Phi_t(\mathbf{z})$。密度函数 $p_t(\mathbf{x})$ 的变换遵循概率守恒定律：尽管分布的几何形状（由等高线图表示）会发生拉伸和压缩，但密度在相应区域上的积分保持不变。

重新整理此项，得到微分同胚下密度变换的标准公式

$$p_t(\mathbf{x}(t)) \quad = \quad p_0(\mathbf{x}(0)) \left|\det \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(0)}\right|^{-1}. \tag{71}$$

这里，$\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(0)}$ 是流映射的雅可比矩阵，它表示从时刻 0 到 t 局部变形的累积。关于 CNF 中概率密度变换的图示，请参见图 10。

对式 (71) 取对数，得到对数密度的变化

$$\log p_t(\mathbf{x}(t)) \quad = \quad \log p_0(\mathbf{x}(0)) - \log \left|\det \frac{\partial \Phi_t(\mathbf{x}(0))}{\partial \mathbf{x}(0)}\right|. \tag{72}$$

这也称为推前测度下的推前密度公式 [Tao, 2011]。

在实践中，计算整个流的雅可比行列式的计算成本很高。我们可以通过考虑无穷小变化来简化这一点。遵循 Chen et al. [2018] 的工作，通过对数密度关于时间求导，我们得到 **瞬时变量替换定理**

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} \quad = \quad -\mathrm{Tr}\left(\frac{\partial f(\mathbf{x}(t), \theta, t)}{\partial \mathbf{x}(t)}\right). \tag{73}$$

这里，$\mathrm{Tr}(\frac{\partial f}{\partial \mathbf{x}})$ 是向量场雅可比矩阵的迹，它等价于散度，记为 $\nabla \cdot f$。这个量描述了无穷小体积的瞬时膨胀或收缩速率。如果散度为正，则体积膨胀，对数密度以等于散度的速率减小。反之，如果散度为负，则体积收缩，密度增加。

这种连续公式化相对于离散归一化流具有计算优势。在离散流中，确保雅可比行列式的易处理性通常需要限制神经网络架构（例如，限制为三角矩阵）[Hoogeboom et al., 2019; Tran et al., 2019]。相比之下，CNF 只需要雅可比矩阵的迹。这允许 $f(\cdot)$ 由任意的深度神经网络参数化，因为与完整行列式的 $O(d^3)$ 成本相比，迹可以使用随机估计器（如 Hutchinson 迹估计器）高效地近似。

在推理时，我们可以积分状态和对数密度的联合动力学。例如，在生成建模中，给定时刻 t_1 的目标数据点 $\mathbf{x}_{\text{data}}$，我们可以通过从 t_1 到 0 反向积分来计算其对数似然

$$\begin{aligned} \log p_{t_1}(\mathbf{x}_{\text{data}}) &= \log p_0(\mathbf{x}(0)) + \int_{t_1}^{0} \mathrm{Tr}\left(\frac{\partial f(\mathbf{x}(t), \theta, t)}{\partial \mathbf{x}(t)}\right) dt \\ &= \log p_0(\mathbf{x}(0)) - \int_{0}^{t_1} \mathrm{Tr}\left(\frac{\partial f(\mathbf{x}(t), \theta, t)}{\partial \mathbf{x}(t)}\right) dt. \end{aligned} \tag{74}$$

这提供了一个框架，其中神经网络学习一个向量场，该向量场将简单的先验分布平滑地变形为复杂的数据分布。这种关于密度演化的连续时间视角是扩散模型等高级生成模型的核心。在下一节中，我们将探讨这些模型如何应用于将噪声转换回高保真数据。

3.3.4 流模型的变体

虽然连续归一化流通过向量场的积分来定义变换，但更早期且更广泛的流模型家族则专注于构建离散的可逆映射序列。这些模型的主要挑战在于设计神经网络，使得雅可比行列式的计算高效（理想情况下与维度呈线性关系，$O(d)$），同时保持较高的表达能力。根据它们如何构建变换和雅可比矩阵，这些模型可以分为几种变体：

- **耦合流**。由 NICE [Dinh et al., 2014] 和 RealNVP [Dinh et al., 2017] 开创，这些模型将输入维度划分为两个子集。一个子集保持不变，另一个则进行由第一个子集参数化的仿射变换。这种构造强制得到一个三角雅可比矩阵，使得行列式计算变得简单（对角元素的乘积）且计算高效。

- **自回归流**。这些模型将输入维度视为一个序列，其中第 i 维的变换仅依赖于前序维度 $\{1, \ldots, i-1\}$。这种结构同样产生一个三角雅可比矩阵。著名的例子包括掩码自回归流（MAF）[Papamakarios et al., 2017]，它对于密度估计是高效的；以及逆自回归流（IAF）[Kingma et al., 2016]，它对于采样是高效的。

- **可逆线性变换**。为了确保所有维度能够相互影响，像 Glow [Kingma and Dhariwal, 2018] 这样的模型引入了可学习的 1×1 卷积。为了保持效率，权重矩阵通常通过 LU 分解进行参数化，将行列式计算的复杂度从三次方 $O(d^3)$ 降低到线性 $O(d)$。

- **残差流**。受残差网络的启发，这些流将映射定义为 $\mathbf{y} = \mathbf{x} + g(\mathbf{x})$。为了保证可逆性，$g(\cdot)$ 的 Lipschitz 常数必须严格有界（通常 < 1）[Behrmann et al., 2019]。与耦合层不同，残差流没有三角雅可比矩阵。相反，它们通常依赖于随机估计器（如俄罗斯轮盘赌估计器）来近似对数行列式的无穷级数展开 [Chen et al., 2019]。

需要注意的是，流是一个非常普遍的概念，在许多领域，特别是在动力系统和微分几何的研究中，已被广泛讨论。从数学上讲，流可以形式化为一个**单参数微分同胚群**。这个定义意味着映射 $\Phi_t(\cdot)$ 满足群律

$$\Phi_0(\mathbf{x}) = \mathbf{x}, \tag{75}$$

$$\Phi_{t+s}(\mathbf{x}) = \Phi_t(\Phi_s(\mathbf{x})). \tag{76}$$

这种代数结构为可逆性提供了理论保证：在时间 t 上的变换的逆，就是在时间 $-t$ 上的变换，即 $\Phi_t^{-1}(\cdot) = \Phi_{-t}(\cdot)$。

此外，从流体力学和统计物理学的角度来看，概率密度的流动由**连续性方程** [Batchelor, 2000] 控制，这是一个描述质量守恒的偏微分方程

$$\frac{\partial p_t(\mathbf{x})}{\partial t} + \nabla \cdot (p_t(\mathbf{x}) f(\mathbf{x}, \theta, t)) = 0. \tag{77}$$

这个偏微分方程表明，某一点处密度的变化率恰好与概率通量的散度相平衡。连续归一化流中使用的瞬时变量变换公式（式 (73)）实际上就是沿着粒子轨迹求解该连续性方程的解。因此，流模型不仅可以被视为具有特定架构约束的神经网络，还可以被视为经典输运现象的计算实现。关于从连续性方程推导瞬时变量变换公式的更详细过程，请参见附录 B。

4. 视觉中的扩散模型

在计算机视觉中，扩散模型指的是一类生成模型，它定义了从简单先验分布到数据分布的参数化变换。一个重要的应用是图像合成，其中样本是通过迭代地对高斯噪声去噪而生成的。尽管扩散模型源于非平衡热力学和变分推断，但它们可以用微分方程优雅地解释。如前面章节所述，残差网络代表了常微分方程的一种简单欧拉离散化，而神经常微分方程则直接对连续时间动力学进行建模。扩散模型将这些概念联系起来。从概念上讲，它们定义了连续时间演化（类似于神经常微分方程/随机微分方程），但在实践中，它们通常使用离散时间步进行优化和部署，类似于深度残差网络。

在本节中，我们首先通过考虑前向和后向变换的双重过程来阐述生成建模的问题。然后，我们介绍建模生成问题的两种视角：随机视角和确定性视角。前者基于**随机微分方程**，将生成过程框定为添加和去除噪声的随机过程。后者基于**概率流**，将生成过程视为确定性过程。最后，我们讨论提高扩散模型效率的方法。

扩散模型已在许多论文 [Croitoru et al., 2023; Yang et al., 2023] 和博客 [Song, 2021; Dieleman, 2023] 中被广泛讨论。本节并不旨在涵盖这些模型的每一个细节，而是侧重于从微分方程的视角来讨论它们。偶尔，我们会稍微扩展讨论以确保内容的完整性。

4.1 问题陈述

在介绍扩散模型之前，我们首先对生成建模问题进行形式化定义。

4.1.1 作为概率传输的生成建模

在生成建模的标准设定中，目标是从数据集学习潜在的概率分布 $p_{\text{data}}(\cdot)$ 并从中生成新样本。例如，如果我们知道自然图像的真实分布，只需从中采样即可生成新图像 $\mathbf{x}$。但由于真实分布未知，通常做法是通过观察到的图像集合进行近似学习。

为实现这一目标，我们通常依赖**概率传输**的概念。由于直接对复杂数据分布 $p_{\text{data}}(\mathbf{x})$ 建模和采样较为困难，我们引入从易处理先验分布 $p_{\text{prior}}(\mathbf{z})$（如标准高斯分布 $\mathcal{N}(\mathbf{0}, \mathbf{I})$）采样的潜变量 $\mathbf{z}$。生成任务因此转化为寻找确定性或随机性映射 $\Phi : \mathbb{R}^d \to \mathbb{R}^d$，将潜变量 $\mathbf{z}$ 转换为数据样本 $\mathbf{x} \approx \Phi(\mathbf{z})$。

从微分方程的角度看，该映射并非瞬时完成，而是由连续时间动态过程诱导产生。在此过程中，$\mathbf{x}$ 和 $\mathbf{z}$ 被视为系统轨迹起点与终点的状态。虽然神经 ODE（见第3节）通常建模任意区间 $[t_0, t_1]$ 上的动态过程，但扩散模型特别地将此传输表述为固定区间 $t \in [0, T]$ 上的连续时间演化。因此我们遵循惯例：过程从 $t = 0$ 的数据分布开始，演化至 $t = T$ 的先验分布，或反之。

形式上，可将这种动态演化视为流。设 $\Phi_{0\to t} : \mathbb{R}^d \to \mathbb{R}^d$ 表示将样本从 0 时刻状态传输至 t 时刻状态的流映射。该映射作用于单个样本时，会诱导概率分布本身的相应变换。这种关系通过**推前**操作数学描述：若初始数据分布为 $p_0(\mathbf{x})$，则任意中间时刻 t 的分布 $p_t(\mathbf{x})$ 是 p_0 经流映射的推前，记为

$$p_t = (\Phi_{0\to t})_{\#} p_0. \tag{78}$$

该等式意味着概率质量在传输过程中守恒：若随机变量 $\mathbf{x}$ 服从 p_0，则变换后的变量 $\Phi_{0\to t}(\mathbf{x})$ 服从 p_t。通常假设该流是微分同胚，因此操作完全可逆。于是生成过程可定义为逆向推前 $p_0 = (\Phi_{T\to 0})_{\#} p_T$，将简单先验 p_T 传输回复杂数据分布 p_0。

为理解推前操作与概率分布演化，可类比将染色剂滴入流动水体的情景。其中概率分布对应染料浓度，数据样本对应染料分子。流映射 $\Phi_{0\to t}$ 可视作水流，决定每个分子随时间推移的精确轨迹。当这些分子随水流运动时，染料云的整体形状发生变化。推前操作是通过分子运动计算特定时空位置染料密度的数学工具，示意图见图11。

需注意，虽然局部密度 $p_t(\mathbf{x})$ 会随流体扩张收缩而变化，但染料总量保持恒定。这意味着概率密度在整个空间上的积分恒等于 1。因此通过定义”粒子”（样本）的流动，我们隐式且唯一地定义了”云团”（分布）的演化。

至此，生成建模的任务明确为：学习逆向流 $\Phi_{T\to 0}$ 及其对应的推前操作 $(\Phi_{T\to 0})_{\#} p_T$，从而将先验噪声样本推前至数据分布。最终输出是从该数据分布抽取的样本。由于 $\Phi_{T\to 0}$ 是流，可用 ODE 或相关方程工具建模。本节后续将介绍几种概率传输建模方法，均可通过微分方程得到合理解释。

虽然流可视为系统状态的单次连续时间演化，但实践中通常将其离散化为多个简单步骤以便建模学习。我们不直接学习噪声与数据分布间的整体映射，而是将过程分解为增量转移序列。具体而言，设 N 为离散化步数，$0 = t_0 < t_1 < \cdots < t_N = T$ 为时间点。从 $\mathbf{z}$ 到 $\mathbf{x}$ 的轨迹表示为 $\mathbf{z} = \mathbf{x}(t_N) \to \mathbf{x}(t_{N-1}) \to \cdots \to \mathbf{x}(t_1) \to \mathbf{x}(t_0) = \mathbf{x}$。在图像生成中，这意味着从纯噪声图像出发，逐步去噪生成真实图像。在特定步骤 k，将分布 p_{t_k} 映射至后续分布 $p_{t_{k-1}}$ 所需的变换较为简单。通过将全局传输问题分解为这些局部转移，模型仅需学习每步流动方向（即向量场）。

4.1.2 前向过程与反向过程的二元性

如上所述，扩散模型设计的核心在于将任务分解为两个过程：一个破坏数据的过程和一个创造数据的过程。我们分别称之为**前向过程**和**反向过程**。

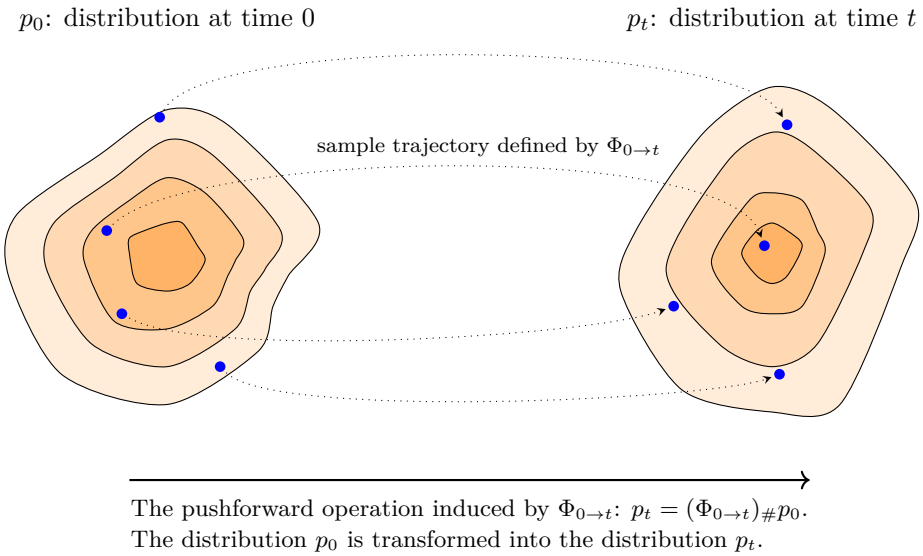


图 11: 推前运算示意图。时间 0 和时间 t 处的分布可视化为流体中染料密度的演化过程。映射 $\Phi_{0\to t}$ 描述了单个染料分子随时间推移的轨迹。当这些分子移动到新位置时，局部密度发生变化，但总质量保持守恒。推前运算形式化地定义了这种变换，它根据 Φ 定义的集体轨迹，将初始密度分布 p_0 映射到结果分布 p_t。

前向过程定义了从数据分布到先验分布的转变，时间从 $t = 0$ 演化到 T。这个过程可以随机地或确定性地表述。这导致了设计扩散模型的两种不同视角，正如本节稍后将介绍的那样。作为一个简单的例子，这里我们通过常微分方程的视角，使用确定性微分方程来描述前向过程。在这种视角下，我们通过一个常微分方程来建模样本轨迹

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t), \tag{79}$$

其中 $f(\mathbf{x}(t), t)$ 是一个向量场，通常是预定义的（例如，高斯噪声注入或线性插值）。给定由该常微分方程定义的流，我们可以使用前推操作将样本的分布从时间 0 变换到任意时间 t（参见公式 (78)）。

反向过程是前向过程的逆过程。如果我们采用常微分方程视角，它对应于动力学的时间反转，从 $t = T$ 演化回 0。给定上述前向常微分方程，生成轨迹由反向时间常微分方程定义

$$\frac{d\mathbf{x}(t)}{dt} = -f(\mathbf{x}(t), t), \tag{80}$$

该方程从 T 到 0 反向求解。注意，理想的反向向量场就是前向场的负值，我们可以使用神经网络来近似它。

这种双向框架自然地与自编码器的架构相吻合 [Kingma and Welling, 2019; Bank et al., 2023]，如图 12 所示。前向过程对应于编码器，它将观测数据 $\mathbf{x}$（即 $\mathbf{x}(0)$）通过一系列中间状态 $\{\mathbf{x}(t)\}$ 映射到潜在变量 $\mathbf{z}$（即 $\mathbf{x}(T)$）。反向过程对应于解码器，它通过反向序列从潜在变量 $\mathbf{z}$ 重建数据 $\mathbf{x}$。

然而，扩散模型与传统自编码器之间存在一些差异：

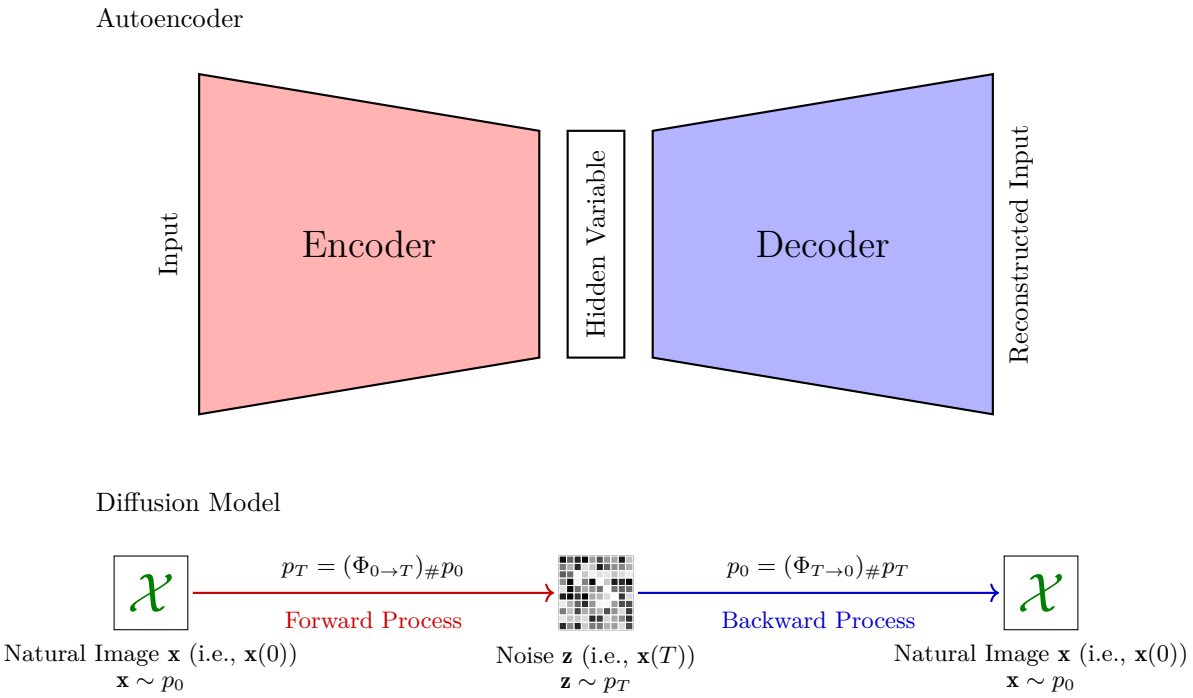


图 12: 自编码器与扩散模型的对比。顶部面板展示了标准自编码器架构，它通过编码器将输入映射到低维隐变量空间，并通过解码器进行重建。底部面板展示了扩散模型。与依赖瓶颈架构压缩数据的自编码器不同，扩散模型定义了数据分布与噪声分布之间的可逆映射。前向过程将数据分布 p_0 推向简单的噪声先验 p_T，而后向过程则将分布映射回 p_0。此处，$\mathbf{x}$ 表示从 p_0 中抽取的自然图像，$\mathbf{z}$ 表示从 p_T 中抽取的噪声。$(\Phi_{0\to T})_{\#}$ 和 $(\Phi_{T\to 0})_{\#}$ 分别表示前向过程和后向过程的推前操作。

- 首先，在维度方面，标准自编码器通常执行降维操作：潜在变量 $\mathbf{z}$ 的维度小于输入 $\mathbf{x}$ 的维度。相比之下，扩散模型在整个过程中通常保持相同的维度。输入数据 $\mathbf{x}$、潜在变量 $\mathbf{z}$ 以及所有中间状态共享相同的维度 $\mathbb{R}^d$。这与我们的任务目标一致——我们旨在转换样本而非改变其维度。
- 其次，关于前向路径与反向路径之间的函数关系，通用自编码器框架并不要求编码器与解码器共享相同的模型架构或参数。它们可以由不同的神经网络分别参数化，这些网络虽联合优化但作为独立函数运作。而扩散模型的概率流常微分方程背景下，前向与反向过程存在本质耦合。如公式 (79) 和 (80) 所示，反向过程由前向向量场的负值所主导。这意味着一种数学对称性：定义前向演化的动力学过程即隐式定义了反向生成过程的动力学。

尽管存在这些差异，自编码器的理论基础为训练扩散模型提供了宝贵的见解 [Luo, 2022; Kingma and Gao, 2023]。例如，**证据下界**，这是**变分自编码器**中常用的损失函数，与训练扩散模型所用的

目标密切相关[7]。在本节中，我们将不讨论自编码器的细节，但会偶尔将其作为解释扩散模型的工具。

4.1.3 训练与推断

虽然前向过程与反向过程在理论上对称，但二者承担着不同的角色。在推断阶段，我们的目标是生成新样本，因此仅执行反向过程。我们首先从先验分布中采样得到潜变量 $\mathbf{z}$（即 $\mathbf{x}(T)$），然后应用学习到的逆向动力学，逐步将噪声映射回数据样本 $\mathbf{x}$（即 $\mathbf{x}(0)$）。

这里自然产生一个问题：如果实际应用仅涉及反向过程，为何需要显式定义前向过程？考虑迷宫行走的类比：若仅从入口走到出口，未观察有效路径时便难以找到归途。前向过程正是构建这些训练路径的引导者——它通过将数据样本 $\mathbf{x}$ 经一系列中间状态转化为噪声 $\mathbf{z}$，为模型提供必要的监督信号。该过程明确告知模型：对于任意给定时间 t，为返回数据分布所需的速度（或方向）应当如何。

因此，扩散模型的训练通常包含两个步骤：首先执行前向过程以生成从数据到噪声的轨迹，从而创建真实路径；随后优化反向过程的参数，使其精确匹配这一真实流形。

4.2 随机视角：随机微分方程

虽然我们已经看到常微分方程为生成建模提供了确定性视角，但重要的是认识到扩散的基本机制本质上是随机的。从这个角度来看，前向过程并非被建模为确定性流，而是被建模为一个用噪声逐步破坏输入样本的随机过程。事实上，这些模型的经典表述依赖于随机微分方程来描述数据破坏和重建的动态过程 [Oksendal, 2013]。接下来，我们将介绍基于随机微分方程的扩散模型。

4.2.1 前向与反向随机微分方程

随机微分方程（SDE）是一种包含一个或多个随机过程项的微分方程。通常，它描述一个同时受到确定性力（可预测的趋势）和随机噪声（不可预测的波动）影响的系统的演化。

为了理解这一点，我们首先回顾描述状态向量 $\mathbf{x}(t)$ 随时间 t 演化的标准常微分方程（ODE）：$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t)$。这里，$f(\cdot)$ 是一个描述状态速度的确定性向量场。给定初始条件 $\mathbf{x}(0)$，未来的轨迹就完全确定了。

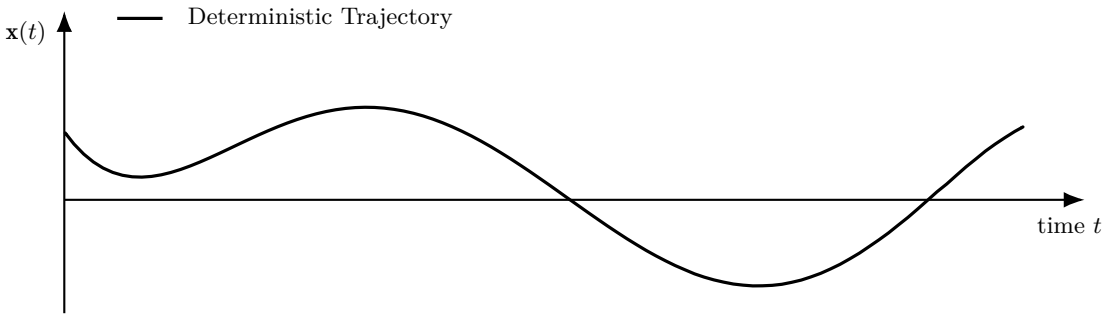
然而，在扩散模型中，我们持续地向数据引入随机噪声。为了形式化这一点，必须在方程中添加一个随机项。由于标准布朗运动处处不可微，我们不能简单地写成 $\frac{d\mathbf{x}(t)}{dt} = f + \text{noise}$。相反，我们以包含随机分量的微分形式写出方程：

$$d\mathbf{x}(t) \quad = \quad \underbrace{f(\mathbf{x}(t), t)dt}_{\text{deterministic}} + \underbrace{g(t)d\mathbf{w}(t)}_{\text{stochastic}}. \tag{81}$$

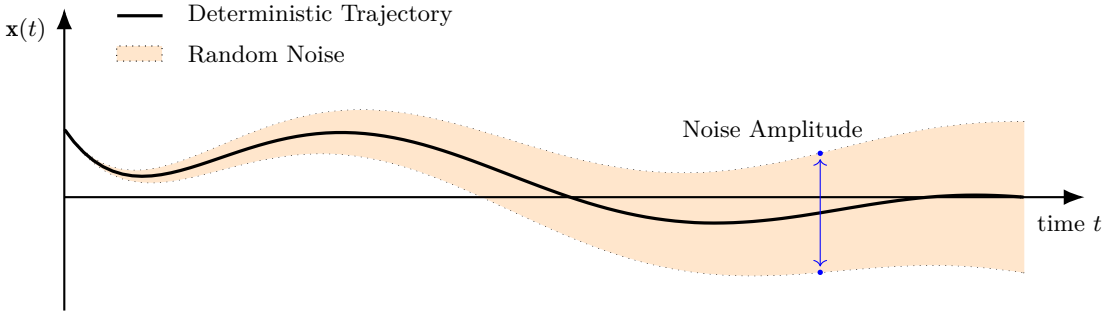
这是一个通用的伊藤（Itô）SDE[8]。该方程的组成部分定义如下：

7. 通过最大化证据下界 [Kingma and Welling, 2013]，我们最小化了每个时间步上前向与反向转移分布之间的散度（通常是 Kullback-Leibler 散度）。这种表述将“反转流”的物理直觉与“最大化数据似然”的统计目标联系起来。因此，生成建模问题被简化为一个回归任务：模型学习估计生成当前状态的向量场，从而学习过程的逆过程。

8. 随机积分 $\int g(t)d\mathbf{w}(t)$ 的计算需要特定的解释。**伊藤解释**在离散化过程中，在每个时间间隔的起点（即左端点）处计算被积函数 [Karatzas and Shreve, 2014]。这种选择在扩散模型中成为标准，因为它确保了噪声增量 $d\mathbf{w}$ 与当前状态独立，从而保持了鞅性质并简化了期望的计算。



(a) ODE (deterministic trajectory)



(b) SDE (with random noise)

图 13: 确定性轨迹与随机轨迹对比。作为对比，子图 (a) 展示了由常微分方程 (ODE) 控制的确定性轨迹。子图 (b) 展示了由 $d\mathbf{x}(t) = f(\mathbf{x}(t), t)dt + g(t)d\mathbf{w}(t)$ 定义的随机微分方程 (SDE) 轨迹。其中 $f(\mathbf{x}(t), t)dt$ 表示确定性漂移项（类比于 ODE），而 $g(t)d\mathbf{w}(t)$ 引入了高斯噪声。在本示例中，漂移系数 $f(\mathbf{x}(t), t)$ 逐渐衰减至零，而扩散系数 $g(t)$ 随时间递增。在终止时刻 T 处满足 $f(\mathbf{x}(T), T) = 0$ 和 $g(T) = 1$，此时状态收敛于标准高斯分布。该行为类似于扩散模型的前向过程：随时间推移，确定性信号逐渐衰减而噪声持续注入。

- 漂移系数 $f(\mathbf{x}(t), t) \in \mathbb{R}^d$ 表示演化的确定性部分，类似于常微分方程中的向量场。它决定了演化的一般趋势。
- 扩散系数 $g(t)$ 通常是一个标量函数，用于控制在时间 t 注入系统的随机噪声的幅度。
- 维纳过程（也称为布朗运动）$\mathbf{w}(t) \in \mathbb{R}^d$ 表示随机性的来源。增量 $d\mathbf{w}(t)$ 可以视为一个均值为 $\mathbf{0}$、协方差为 $dt \cdot \mathbf{I}$ 的无穷小高斯噪声向量。

方程 (81) 被称为前向 SDE。通过使用这个 SDE，我们定义了一个前向过程，当时间 t 从 0 流向 T 时，该过程逐渐向数据样本 $\mathbf{x}(0) \sim p_{\text{data}}$ 添加噪声。在此过程中，分布 $p_t(\mathbf{x}(t))$ 扩散，并且 $\mathbf{x}(T)$ 的分布最终将收敛到一个已知的先验分布（通常是 $\mathcal{N}(\mathbf{0}, \mathbf{I})$），如图 13 所示。

SDE 的一个显著特性，由 Anderson [1982] 推导得出，是对于任何前向 SDE，都存在一个对应的反向时间 SDE。如果我们从 $t = T$ 到 0 反向模拟这个过程，就可以从数据分布中生成样本。

反向 SDE 由下式给出：

$$d\mathbf{x}(t) \quad = \quad \left[f(\mathbf{x}(t),t) - g(t)^2 \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))\right] dt + g(t) d\bar{\mathbf{w}}(t), \tag{82}$$

其中 dt 表示一个无穷小的负时间增量，$d\bar{\mathbf{w}}(t)$ 是反向时间中的一个标准维纳过程，而 $\nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))$ 是概率密度对数相对于数据的梯度，也称为 **得分函数**。虽然这个方程看起来有点复杂，但其应用是直接的。给定一个前向 SDE（方程 (81)），如果我们能够访问得分函数，就可以使用反向 SDE（方程 (82)）将样本从先验噪声 p_T 传输回数据分布 p_0。

将反向 SDE 应用于生成建模时存在两个基本问题。

- **如何获取反向 SDE 中的得分函数？**反向过程依赖于对数密度的梯度 $\nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))$。然而，中间边缘分布 $p_t(\mathbf{x}(t))$ 是通过对所有可能的数据样本和扩散路径进行边缘化得到的，这使得其解析求解不可行。我们只能获取数据分布 p_0 的样本，而无法获得 p_t 的显式表达式。
- **如何定义正向 SDE？**我们需要指定漂移系数 $f(\mathbf{x}(t),t)$ 和扩散系数 $g(t)$。这些函数决定了扩散过程的轨迹。必须精心设计这些函数，以确保复杂的数据分布 p_0 在过程结束时能有效地转化为简单的先验分布 p_T。

接下来，我们将通过引入 **得分匹配**技术并讨论文献中常用的 SDE 公式来解决这些问题。

4.2.2 学习分数函数

如前所述，由于边缘分布 $p_t(\mathbf{x}(t))$ 通常是难以处理的，我们无法直接计算梯度 $\nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))$。因此，我们转而采用分数匹配方法 [Hyvärinen and Dayan, 2005]，特别是**去噪分数匹配**技术 [Vincent, 2011]。

其核心思想是，虽然边缘分布的分数未知，但条件转移分布 $p(\mathbf{x}(t)|\mathbf{x}(0))$ 的分数通常易于计算。具体来说，考虑一个转移核，其中将高斯噪声 $\boldsymbol{\epsilon}$ 添加到数据点 $\mathbf{x}(0)$ 上，使得 $\mathbf{x}(t) \sim \mathcal{N}(\mathbf{x}(0), \sigma(t)^2\mathbf{I})$，即：

$$\mathbf{x}(t) \quad = \quad \mathbf{x}(0) + \boldsymbol{\epsilon}. \tag{83}$$

该条件分布的分数由下式给出：

$$\begin{aligned} \nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t)|\mathbf{x}(0)) \quad &= \quad -\frac{\mathbf{x}(t) - \mathbf{x}(0)}{\sigma(t)^2} \\ &= \quad -\frac{\boldsymbol{\epsilon}}{\sigma(t)^2}. \end{aligned} \tag{84}$$

Vincent [2011] 证明，训练一个神经网络 $s_\theta(\mathbf{x}(t),t)$ 来近似这个条件分数（与所添加的噪声 $\boldsymbol{\epsilon}$ 成正比），等价于学习边缘分数 $\nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))$。因此，生成建模的目标可以有效地简化为一个去噪回归问题。

形式上，这一直觉引出了以下目标函数。我们希望最小化估计分数与真实条件分数之间的期望平方欧几里得距离：

$$\mathcal{L}(\theta) \quad = \quad \mathbb{E}_{t,\mathbf{x}(0),\boldsymbol{\epsilon}} \left[\lambda(t) \left\| s_\theta(\mathbf{x}(t),t) - \left(-\frac{\boldsymbol{\epsilon}}{\sigma(t)^2} \right) \right\|^2 \right], \tag{85}$$

其中 $\lambda(t)$ 是一个正权重函数。一个常见的选择是 $\lambda(t) = \sigma(t)^2$，这可以平衡不同噪声水平下分数的大小。

通过代入此权重并重新整理项，目标简化为：

$$\mathcal{L}(\theta) \quad = \quad \mathbb{E}_{t,\mathbf{x}(0),\boldsymbol{\epsilon}} \left[\frac{1}{\sigma(t)^2} \left\| \boldsymbol{\epsilon} + \sigma(t)^2 s_\theta(\mathbf{x}(0) + \boldsymbol{\epsilon}, t) \right\|^2 \right]. \tag{86}$$

这个公式表明，优化分数模型在数学上等价于训练一个网络来预测添加到样本中的噪声 $\boldsymbol{\epsilon}$。在实践中，权重因子 $1/\sigma(t)^2$ 通常被舍弃以优先考虑感知质量，这导致了一个简单的噪声预测目标。

值得注意的是，上述方法与**基于能量的模型**（EBMs）[LeCun et al., 2006] 存在一些理论联系。在统计物理学中，概率分布通常通过能量函数 $E_\theta(\mathbf{x})$ 表示为 $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$，其中 Z_θ 是归一化常数或配分函数。

对分数函数而非似然本身进行建模的一个优点是，它可以绕开难以处理的归一化常数 Z_θ 的计算。要使一个模型归一化，其密度在整个空间上的积分必须等于 1。例如，语言模型使用 Softmax 函数来归一化逻辑值，这需要对词汇表进行求和。然而，对于像图像这样的高维连续数据，计算 $Z_\theta = \int \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$ 在计算上是难以处理的。基于分数的模型避免了这个问题，因为对数配分函数关于数据的梯度为零，即 $\nabla_\mathbf{x} \log Z_\theta = 0$。因此，我们有：

$$\begin{aligned} \nabla_\mathbf{x} \log p_\theta(\mathbf{x}) \quad &= \quad \nabla_\mathbf{x} \left(-E_\theta(\mathbf{x}) - \log Z_\theta \right) \\ &= \quad -\nabla_\mathbf{x} E_\theta(\mathbf{x}). \end{aligned} \tag{87}$$

这个特性允许我们在不评估配分函数的情况下训练未归一化的模型。

一旦分数网络 $s_\theta(\mathbf{x}, t)$ 训练完成，我们就可以用它来生成新样本。具体来说，我们将学习到的近似值 $s_\theta(\mathbf{x}(t), t) = \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))$ 代入方程 (82)，得到：

$$d\mathbf{x}(t) = \left[f(\mathbf{x}(t), t) - g(t)^2 s_\theta(\mathbf{x}(t), t) \right] dt + g(t) d\bar{\mathbf{w}}(t). \tag{88}$$

在测试时，我们从先验分布（例如 $\mathcal{N}(\mathbf{0}, \mathbf{I})$）初始化 $\mathbf{x}(T)$，并使用欧拉-丸山方法 [Kloeden and Pearson, 1977] 等数值求解器，从 $t = T$ 到 $t = 0$ 反向积分该方程。通过逐步去除噪声，这个过程将初始的噪声样本转化为目标数据分布的样本。

4.2.3 VE 与 VP 随机微分方程

我们现在转向定义前向随机微分方程（SDE）的问题，即在方程 (81) 中定义漂移系数 $f(\mathbf{x}(t), t)$ 和扩散系数 $g(t)$。构建前向 SDE 有多种方法，这里我们考虑两种常用方法，它们分别对应两个离散时间模型的连续时间极限：**噪声条件得分网络**（NCSN）[Song and Ermon, 2019] 和**去噪扩散概率模型**（DDPM）[Ho et al., 2020b]。Song et al. [2021] 将这些方法统一在 SDE 框架下，并命名为**方差爆炸**（VE）和**方差保持**（VP）SDE。

在 NCSN 框架中，前向过程通过一系列递增的高斯噪声水平 $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$（其中 $\sigma_1 < \sigma_2 < \cdots < \sigma_N$）扰动数据。考虑一个离散过程，其中第 i 步的状态 $\mathbf{x}_i$ 通过从 $\mathbf{x}_{i-1}$ 添加必要噪声量使总方差从 σ_{i-1}^2 增加到 σ_i^2 演化而来，其更新规则为：

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\boldsymbol{\epsilon}_i, \tag{89}$$

其中 $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 表示第 i 步的增量噪声。该模型确保若从数据 $\mathbf{x}_0$ 出发，第 i 步的分布将具有 σ_i^2 的方差[9]。

现在转向连续极限。我们定义关于时间 t 的连续方差函数 $\sigma(t)$。当步数趋近无穷大（$N \to \infty$）时，离散时间间隔趋近于零（$\Delta t \to 0$）。相邻步间的方差差变为微分：

$$\sigma_i^2 - \sigma_{i-1}^2 \approx \frac{d\sigma^2(t)}{dt}\Delta t. \tag{90}$$

回忆随机微积分中，维纳过程增量 $d\mathbf{w}$ 与 $\sqrt{\Delta t}$ 成比例（即 $\sqrt{\Delta t}\boldsymbol{\epsilon}_i \approx d\mathbf{w}(t)$）。通过替换方差差可重写更新规则：

$$\mathbf{x}_i - \mathbf{x}_{i-1} \approx \sqrt{\frac{d\sigma^2(t)}{dt}\Delta t}\,\boldsymbol{\epsilon}_i. \tag{91}$$

取极限 $\Delta t \to 0$ 后，我们得到连续时间 SDE：

$$d\mathbf{x}(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}d\mathbf{w}(t). \tag{92}$$

该 SDE 中漂移系数定义为 $f(\mathbf{x}(t), t) = \mathbf{0}$，扩散系数为 $g(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}$。由于分布方差随 $t \to T$ 单调递增，此 SDE 将数据分布映射至具有发散方差的噪声分布，因此称为方差爆炸 SDE。

另一方面，DDPM 框架采用不同的噪声注入策略。不同于简单地增加方差噪声，它在每一步重新缩放数据以保持方差有界。在离散公式中，前向过程由方差调度 $0 < \beta_1 < \beta_2 < \cdots < \beta_N < 1$ 定义，从 $\mathbf{x}_{i-1}$ 到 $\mathbf{x}_i$ 的转移由下式给出：

$$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\boldsymbol{\epsilon}_i, \tag{93}$$

其中 $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$。与 NCSN 公式不同，系数 $\sqrt{1-\beta_i}$ 会收缩前一状态的均值。该设计确保若输入数据服从标准正态分布，则每步边缘分布仍保持标准正态（即方差得以保持）。

此时增量 $\mathbf{x}_i - \mathbf{x}_{i-1}$ 可表示为：

$$\mathbf{x}_i - \mathbf{x}_{i-1} = (\sqrt{1-\beta_i} - 1)\mathbf{x}_{i-1} + \sqrt{\beta_i}\boldsymbol{\epsilon}_i. \tag{94}$$

为获得连续时间 SDE，我们再次令 $N \to \infty$ 并引入连续函数 $\beta(t)$ 使得 $\beta_i \approx \beta(t)\Delta t$。利用泰勒展开近似 $\sqrt{1-x} \approx 1 - \frac{1}{2}x$（当 x 较小时），有 $\sqrt{1-\beta(t)\Delta t} - 1 \approx -\frac{1}{2}\beta(t)\Delta t$。将其代入式 (94) 并应用缩放规则 $\sqrt{\beta(t)\Delta t}\boldsymbol{\epsilon}_i \approx \sqrt{\beta(t)}d\mathbf{w}(t)$，可得极限：

$$d\mathbf{x}(t) = -\frac{1}{2}\beta(t)\mathbf{x}(t)dt + \sqrt{\beta(t)}d\mathbf{w}(t). \tag{95}$$

此 SDE 中漂移系数为 $f(\mathbf{x}(t), t) = -\frac{1}{2}\beta(t)\mathbf{x}(t)$，扩散系数为 $g(t) = \sqrt{\beta(t)}$。漂移项作为恢复力将状态拉向原点（零均值），从而抵消扩散项注入的方差。因此若初始分布具有单位方差，整个过程方差将保持恒定。故式 (95) 称为方差保持 SDE。

9. 更准确地说，由于初始状态 $\mathbf{x}_0$ 给定，我们应说明在给定 $\mathbf{x}_0$ 条件下分布的方差为 σ_i^2

Entry	Variance Exploding (NCSN)	Variance Preserving (DDPM)
Drift coefficient $f(\mathbf{x}(t), t)$	$\mathbf{0}$	$-\frac{1}{2}\beta(t)\mathbf{x}(t)$
Diffusion coefficient $g(t)$	$\sqrt{\frac{d\sigma^2(t)}{dt}}$	$\sqrt{\beta(t)}$
Continuous-time SDE	$d\mathbf{x}(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}d\mathbf{w}(t)$	$d\mathbf{x}(t) = -\frac{1}{2}\beta(t)\mathbf{x}(t)dt + \sqrt{\beta(t)}d\mathbf{w}(t)$
Iterative diffusion (discrete)	$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\boldsymbol{\epsilon}_i$	$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\boldsymbol{\epsilon}_i$
Single-step diffusion (discrete)	$\mathbf{x}_i = \mathbf{x}_0 + \sigma_i\boldsymbol{\epsilon}$	$\mathbf{x}_i = \sqrt{\bar{\alpha}_i}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}$

表 3: VP 和 VE SDE 的连续时间与离散时间形式 [Song et al., 2021]。$\mathbf{x}(t)$、$\mathbf{w}(t)$、$\sigma(t)$ 和 $\beta(t)$ 是时间 t 的连续函数。$\mathbf{x}_i$ 是离散时间步 i 的状态，β_i、σ_i 和 $\bar{\alpha}_i$ 是对应的参数。此处 β_i 和 σ_i 是预先指定的，$\bar{\alpha}_i$ 定义为 $\prod_{s=1}^{i}(1-\beta_s)$，它决定了单步扩散过程中信号 $\mathbf{x}_0$ 的相对权重。$\boldsymbol{\epsilon}_i$ 和 $\boldsymbol{\epsilon}$ 均为标准高斯噪声。前者表示第 i 步的增量噪声，后者表示将 $\mathbf{x}_0$ 直接扰动至 $\mathbf{x}_i$ 的总噪声。

为简洁起见，此处未明确定义参数 $\{\sigma_i\}$（或 $\sigma(t)$）和 $\{\beta_i\}$（或 $\beta(t)$）的函数形式。值得注意的是，DDPM 中常引入辅助变量（如 $\alpha_i = 1-\beta_i$ 和 $\bar{\alpha}_i = \prod_{s=1}^{i}\alpha_s$）以简化推导与实现。调度参数的精心设计可使模型理论完备。关于这些参数选择的具体讨论，建议读者参阅原始论文 [Song and Ermon, 2019; Ho et al., 2020b]。

虽然本节重点是通过连续时间 SDE 公式建立更统一的理论基础，但需注意实践中扩散模型通常通过离散化步骤实现（见表3）。NCSN 和 DDPM 方法中引入的离散变量 $\mathbf{x}_i$ 本质上是连续过程的数值近似。实际实现中，我们通过迭代这些离散时间步 $\{\mathbf{x}_i\}$ 来模拟 SDE 演化。这反映了微分方程应用于深度学习的常见范式：利用连续微分方程的理论完备性和表达能力，同时依赖离散数值近似实现高效训练与推理。

4.2.4 数值求解器

为了从训练好的扩散模型中生成样本，我们需要从初始样本 $\mathbf{x}(T) \sim p_T$ 出发，逆向模拟反向时间 SDE（式 (82)）直至 $t = 0$。由于复杂数据分布通常无法获得闭式解，我们采用数值求解器来近似连续轨迹。

最基础且广泛使用的方法是 Euler-Maruyama 方法，该方法将欧拉法推广至 SDE 情形 [Kloeden and Platen, 1992]。我们首先将时间区间 $[0, T]$ 离散化为 N 步：$t_N = T > t_{N-1} > \cdots > t_0 = 0$。对于微小时间步长 $\Delta t = t_{i+1} - t_i$，反向 SDE 可通过下式近似：

$$\mathbf{x}_{t_i} \approx \mathbf{x}_{t_{i+1}} - \left[f(\mathbf{x}_{t_{i+1}}, t_{i+1}) - g(t_{i+1})^2 s_\theta(\mathbf{x}_{t_{i+1}}, t_{i+1})\right]\Delta t + g(t_{i+1})\sqrt{\Delta t}\,\boldsymbol{\epsilon}_{i+1}, \tag{96}$$

需要注意的是，由于我们进行的是时间逆向积分，数值求解器中的计算步骤对应于物理时间 t 的负增量。该公式化方法重现了原始 NCSN 和 DDPM 论文中使用的采样步骤，并为其启发式采样算法提供了理论依据[10]。

10. 例如，将 Euler-Maruyama 离散化应用于反向 VP SDE（从 t_{i+1} 到 t_i）可得 $\mathbf{x}_{t_i} \approx (1 + \frac{1}{2}\beta_{i+1})\mathbf{x}_{t_{i+1}} + \beta_{i+1}s_\theta(\mathbf{x}_{t_{i+1}}, t_{i+1}) + \sqrt{\beta_{i+1}}\boldsymbol{\epsilon}_{i+1}$。通过代入分数参数化 $s_\theta(\mathbf{x}, t) \approx -\boldsymbol{\epsilon}_\theta(\mathbf{x}, t)/\sqrt{1-\bar{\alpha}_t}$（其中下标对应 t_{i+1}），并

SDE 框架的优势在于能够将数值积分与**马尔可夫链蒙特卡洛**（MCMC）方法结合 [Robert et al., 1999]。例如，通过使用预测-校正方法，可将数值离散化与基于分数的校正解耦。在预测步骤中，数值求解器（如 Euler-Maruyama 或更高阶求解器）根据当前状态 $\mathbf{x}_{t_{i+1}}$ 估计下一时间步 $\mathbf{x}_{t_i}$ 的状态，该步骤使过程沿时间逆向演化。在预测步骤之后，对 $\mathbf{x}_{t_i}$ 应用 MCMC 方法（如朗之万动力学）进行多次迭代。由于分数函数 $s_\theta(\mathbf{x}_{t_i}, t_i)$ 定义了分布 p_{t_i}，运行朗之万动力学可将预测样本推向边际分布 p_{t_i} 的高密度区域，同时保持时间 t_i 不变。

4.3 确定性视角：概率流常微分方程

生成建模的另一种视角将生成任务框架为由常微分方程定义的概率流。与随机视角不同，这种方法提供了概率传输的确定性表述，使得能够通过连续时间的常微分方程轨迹，在数据分布与先验分布之间建立双向映射。在本小节中，我们首先论证扩散随机微分方程与这些确定性流之间的理论联系。然后，我们讨论直接学习这些确定性流的方法，例如**流匹配**。

4.3.1 从随机微分方程到常微分方程

基于分数的生成模型中一个有趣的理论结果是：随机微分方程（SDE）描述的随机过程具有等价的确定性表示。Song et al. [2021] 证明，对于任意扩散型随机微分方程，都存在一个对应的确定性常微分方程（ODE），其轨迹产生的边缘概率密度 $\{p_t(\mathbf{x}(t))\}_{t=0}^T$ 与随机微分方程完全相同。

该常微分方程被称为概率流常微分方程，其表达式为：

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t)). \tag{97}$$

推导过程依赖于福克-普朗克方程（又称柯尔莫哥洛夫正向方程），该方程描述了随机微分方程动力学下概率密度函数 $p_t(\mathbf{x})$ 的时间演化。可以证明，式 (97) 中常微分方程导出的连续性方程与式 (81) 中随机微分方程的福克-普朗克方程完全一致（更详细的推导见附录 C）。因此，如果我们采样 $\mathbf{x}(T) \sim p_T$ 并逆向求解该常微分方程（从 T 到 0），所得样本 $\mathbf{x}(0)$ 将服从数据分布 p_0，这与逆向随机微分方程的结果完全相同。

这种等价性为随机视角与确定性视角建立了重要联系。需注意式 (97) 依赖于分数函数 $\nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t))$。这意味着一旦训练好分数模型 $s_\theta(\mathbf{x}(t), t)$，即可直接用于构建概率流常微分方程：

$$\frac{d\mathbf{x}(t)}{dt} \approx f(\mathbf{x}(t), t) - \frac{1}{2} g(t)^2 s_\theta(\mathbf{x}(t), t). \tag{98}$$

在实际应用中，相比随机微分方程方法，采用常微分方程进行采样具有多项优势。首先，给定固定初始噪声向量 $\mathbf{x}(T)$ 时，生成过程是确定性的。因此我们可获得唯一的流映射，并能通过正向积分常微分方程将真实图像反演为对应的潜在噪声表示。其次，常微分方程的轨迹通常比随机微分方程更平滑且更易求解。相较于随机微分方程所需的欧拉-丸山方法，我们可以采用具有自适应步长的先进黑盒常微分方程求解器（如龙格-库塔方法），以更少步骤生成高质量样本。第三，利用瞬时变量变换公式，可以精确计算概率流常微分方程下数据的对数似然，这为模型性能的定量评估提供了便利。

注意到 $1 + \frac{1}{2}\beta_{i+1}$ 对应于 DDPM 中使用的缩放项 $(1-\beta_{i+1})^{-1/2}$ 的一阶泰勒展开，所得更新规则即与标准 DDPM 采样算法完全一致。

4.3.2 流匹配

虽然从随机微分方程导出的概率流常微分方程提供了确定性的生成过程，但它本质上是随机扩散过程的副产品。方程 (97) 中的向量场 $f(\mathbf{x}(t),t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}(t)}\log p_t(\mathbf{x}(t))$ 由前向扩散随机微分方程（例如方差爆炸或方差保持）的具体选择决定。这引发了一个问题：我们能否直接学习概率流常微分方程的向量场，而不依赖于中间的扩散公式？此外，我们能否设计这个向量场使其具有理想的特性，例如更直的轨迹，从而更容易进行数值积分？

这催生了流匹配框架 [Lipman et al., 2023; Liu et al., 2023; Albergo and Vanden-Eijnden, 2023]。流匹配是一种无模拟训练连续归一化流的方法，它允许灵活地定义概率路径。

流匹配的核心思想是直接用一个神经网络向量场 $v(\mathbf{x}(t),\theta,t)$（或 $v_\theta(\mathbf{x}(t),t)$）去回归一个能够生成期望概率路径 $p_t(\mathbf{x}(t))$ 的目标向量场 $u_t(\mathbf{x}(t))$。在此，我们将基于流匹配的方法与基于得分的方法进行比较，如下所示：

- **基于分数的方法**。前向过程是一个固定的高斯扩散过程。如前面小节所示，对应的概率流常微分方程具有一个由线性漂移项和分数项组成的向量场。由此产生的轨迹通常波动较大，因为扩散过程按照固定的调度（例如指数衰减的信号）破坏信息。因此，求解反向常微分方程需要许多步数或复杂的求解器来精确追踪这些波动路径。

- **流匹配方法**。与由随机微分方程决定动力学不同，流匹配允许我们设计路径。我们明确定义了从噪声分布中的样本如何映射到数据分布中的样本。例如，两点之间最简单的传输映射是一条直线。通过强制轨迹为直线，向量场沿路径在时间上变为常数。在推理过程中，这样的常微分方程更容易求解。

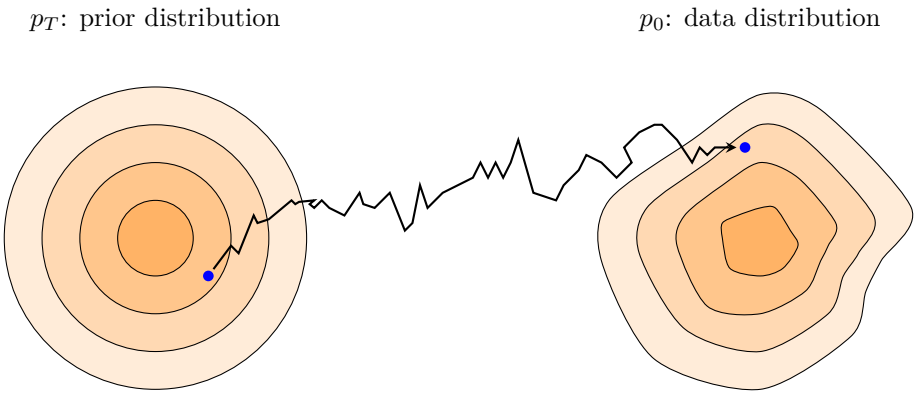
图 14 说明了这种差异。扩散模型在噪声和数据之间找到一条弯曲的路径，而流匹配则旨在找到一条更高效的路径，能够更直接地到达目标数据。

为了与本文使用的符号保持一致，我们将标准的流匹配公式（通常定义为从 $t=0$ 的噪声到 $t=1$ 的数据，如 [Liu et al., 2023]）调整到我们的时间设定，其中 $t=0$ 对应数据分布 p_0，$t=T$ 对应噪声先验 p_T。

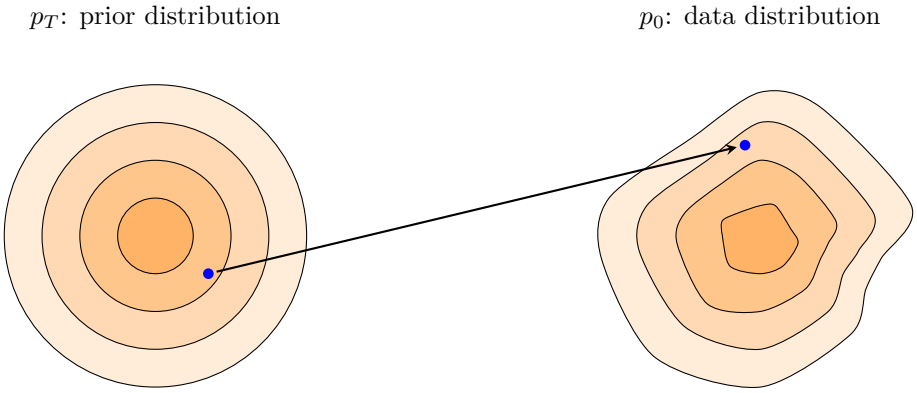
在随机微分方程框架中，概率路径 $p_t(\mathbf{x}(t))$ 是由扩散过程隐式定义的。如果不模拟随机微分方程或计算复杂的边缘分布，我们无法知道中间样本 $\mathbf{x}(t)$ 的闭式表达式。流匹配翻转了这一逻辑：我们显式地构造路径。流匹配的关键洞见在于处理条件概率路径。我们不直接建模难以处理的边缘向量场 $u_t(\mathbf{x}(t))$，而是将路径条件化于一个特定的数据样本 $\mathbf{x}(0)\sim p_0$ 和一个特定的噪声样本 $\mathbf{x}(T)\sim p_T$。我们定义一个连接它们的条件流 $\psi_t(\mathbf{x}(t)|\mathbf{x}(0),\mathbf{x}(T))$。一个简单而有效的选择是线性插值：

$$\begin{aligned}\mathbf{x}(t) &= \psi_t(\mathbf{x}(t)|\mathbf{x}(0),\mathbf{x}(T)) \\ &= \left(1-\frac{t}{T}\right)\mathbf{x}(0)+\frac{t}{T}\mathbf{x}(T),\end{aligned} \tag{99}$$

这个方程通过纯粹的几何方式而非随机扩散步骤定义了一个前向过程。我们可以将其解释为在高维空间中连接一个数据点和一个噪声点的一条直线。该路径的速度就是 $\mathbf{x}(t)$ 对时间的导数，由下



(a) Trajectory of the score-based approach



(b) Trajectory of the flow matching approach (straight path transport)

图 14: 从先验噪声分布（p_T）到数据分布（p_0）的生成轨迹对比。子图（a）展示了基于分数的方法（通过 SDE）生成的随机轨迹。该过程涉及随机噪声注入，导致轨迹呈锯齿状，通常需要大量离散化步骤。子图（b）展示了流匹配方法构建的向量场，它将样本沿直线路径传输。此处我们展示了理想情况下单步生成的过程。尽管在实践中通常需要更多的生成步骤，但这种线性特性使得生成效率相比 SDE 路径显著提高。

式给出：

$$\begin{aligned} u_t(\mathbf{x}(t)|\mathbf{x}(0),\mathbf{x}(T)) &= \frac{d\mathbf{x}(t)}{dt} \\ &= \frac{1}{T}(\mathbf{x}(T)-\mathbf{x}(0)). \end{aligned} \tag{100}$$

注意，对于特定的配对 $(\mathbf{x}(0), \mathbf{x}(T))$，这个目标向量场相对于时间是恒定的，这意味着轨迹是直的。这种路径设计通常被称为 **最优传输**条件向量场，因为它代表了在平方欧几里得成本下将质量从一个分布传输到另一个分布的最有效（最短）路径。

从形式化的角度来看，这种建模方法提供了一个显著的优势：它消除了定义前向动力学时对随机扩散过程的依赖。相反，从 $t = 0$ 到 T 的前向轨迹直接通过方程 (99) 中的解析插值获得。这使得我们能够高效地生成任意时间 t 的中间状态 $\mathbf{x}(t)$，而无需迭代模拟的计算开销。这些插值样本及其对应的目标速度，直接作为训练 “反向” 模型的真实值。具体来说，神经网络向量场 $v_\theta(\mathbf{x}(t), t)$ 被优化以回归这个目标速度 u_t，从而学习反转该流以进行生成。

基于上述公式，我们可以通过将全局向量场 $v_\theta(\mathbf{x}, t)$ 回归到条件向量场来学习它。Lipman et al. [2023] 证明了最小化模型输出与条件向量场之间的差异，等价于最小化其与真实边缘向量场之间的差异。

训练目标，称为 **条件流匹配**，定义为：

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,\mathbf{x}(0),\mathbf{x}(T)} \left[\| v_\theta(\mathbf{x}(t), t) - u_t(\mathbf{x}(t)|\mathbf{x}(0), \mathbf{x}(T)) \|^2 \right]. \tag{101}$$

在实践中，训练过程相当简单：

训练流匹配模型：

1. 采样一个数据点 $\mathbf{x}(0) \sim p_{\text{data}}$ 和一个噪声点 $\mathbf{x}(T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$。
2. 采样一个时间 $t \sim \mathcal{U}[0, T]$。
3. 通过线性插值（公式 (99)）计算中间状态 $\mathbf{x}(t)$。
4. 计算目标速度 $u_t = (\mathbf{x}(T) - \mathbf{x}(0))/T$。
5. 更新网络 v_θ 以最小化 $\|v_\theta(\mathbf{x}(t), t) - u_t\|^2$。

直观上，在训练过程中，模型看到许多连接数据和噪声的交叉直线。通过考虑所有这些条件路径，网络学习到一个连贯的全局向量场，该场以最直接的方式将噪声推向数据区域（反之亦然）。这引出了关于学习到的向量场性质的一个重要理论见解。由于目标函数使用均方误差，优化后的模型 v_θ 不会记忆单个轨迹。相反，它收敛到给定当前状态时目标向量场的条件期望。从数学上讲，最优向量场 $v^*(\mathbf{x}, t)$，称为 **边缘向量场**，满足：

$$v^*(\mathbf{x}, t) = \mathbb{E}_{p(\mathbf{x}(0),\mathbf{x}(T)|\mathbf{x}(t))} \left[u_t(\mathbf{x}(t)|\mathbf{x}(0), \mathbf{x}(T)) \right]. \tag{102}$$

这意味着在任何特定位置 $\mathbf{x}(t)$ 和时间 t，学习到的向量是经过该点的所有可能直线轨迹瞬时速度的 **统计平均值**，如图 15 所示。因此，学习到的全局向量场生成了边缘概率路径 $p_t(\mathbf{x})$，该路径将整个噪声分布传输到数据分布。

一旦模型 $v_\theta(\mathbf{x}, t)$ 被训练好以近似从 $t = 0$（数据）指向 $t = T$（噪声）的向量场，生成过程就通过反向求解常微分方程来完成。我们从纯噪声 $\mathbf{x}(T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 开始，积分到 $t = 0$。由于我们是反向积分，数值更新使用场的负方向：$\mathbf{x}(t - \Delta t) \approx \mathbf{x}(t) - v_\theta(\mathbf{x}(t), t)\Delta t$。

与标准扩散模型相比，这种方法的主要优势在于轨迹形状。使用最优传输路径的流匹配鼓励学习到的全局向量场尽可能直。由此产生的直轨迹对于常微分方程求解器来说更容易跟踪，并且

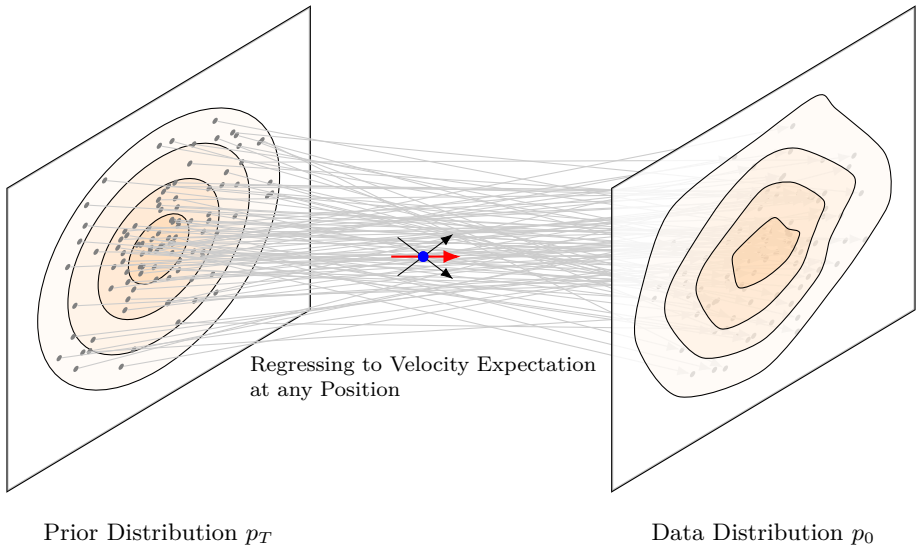


图 15: 流匹配中的边缘向量场学习示意图。灰色线条表示连接噪声样本 $\mathbf{x}(T)$ 到数据样本 $\mathbf{x}(0)$ 的线性插值轨迹。在训练过程中，在特定位置 $\mathbf{x}(t)$（蓝色圆点）处，可能有多个轨迹经过同一点。由于训练目标是最小化均方误差，学习到的向量场 $v_\theta(\mathbf{x}, t)$ 不会记忆单个路径，而是收敛到目标速度的条件期望（红色箭头）。

我们可以采用更大的步长 Δt。因此，与基于随机微分方程的扩散模型相比，流匹配模型通常可以用更少的步数生成高质量的样本。

4.4 迈向更高效的生成

将扩散模型部署到实际应用中的一个瓶颈是其在推理阶段的高计算成本。高效的扩散建模本身就是一个广泛的研究主题 [Shen et al., 2025]。在此，我们简要概述近期关于高效生成的方法，包括高阶数值求解器、轨迹校正和一致性模型。

4.4.1 高阶与专用求解器

最直接的加速方法之一是采用高阶数值求解器。然而，当轨迹弯曲时，像一阶欧拉法（例如扩散上下文中的 DDIM 采样）这样的标准求解器需要许多步才能最小化离散化误差。为了解决这个问题，人们利用扩散常微分方程特有的数学结构，采用了高阶求解器。

例如，DPM-Solver 和 DPM-Solver++ 利用扩散常微分方程的半线性结构来解析计算解的线性部分，从而将非线性部分简化为指数加权积分 [Lu et al., 2022]。其核心原理在于常数变易公式，该公式将常微分方程分离为线性漂移项和非线性基于分数的项。通过解析求解线性部分，求解器只需使用 $(k-1)$ 阶多项式来近似非线性分数函数。DPM-Solver++ 通过使用数据预测参数化和动态阈值处理来防止解漂移到有效数据范围之外，从而进一步提高了稳定性 [Lu et al., 2025]。最近，Zheng et al. [2023] 提出了 DPM-Solver-v3，其特点是采用预测器-校正器框架，通过内部预测步骤来细化积分近似。结合通过经验模型统计优化的系数，该方法在少步数（例如 < 10 步）情况下实现了卓越的性能，且无需额外的函数评估。

除了通用的数值方法外，从几何和数学角度还发展出了几种独特的方法。例如，近似平均方向求解器（AMED-Solver）利用了采样轨迹通常位于低维子空间这一观察结果 [Zhou et al., 2024]。通过学习预测流的平均方向，它可以在更少的步数内收敛。类似地，基于指数积分器，扩散指数积分采样器（DEIS）通过在变换后的对数 ρ 空间中拟合多项式来减少离散化误差 [Zhang and Chen, 2022]。这种变换比标准时间步长更有效地线性化了采样轨迹的曲率。另一种方法是统一预测器-校正器方法（UniPC），它引入了一个自由的校正步骤，重用预测器步骤中已经评估过的噪声预测 [Zhao et al., 2023]。通过重用模型输出，该方法提高了精度阶数，而无需额外的神经函数评估。扩散模型的伪数值方法（PNDM）将去噪过程视为在流形上求解微分方程，该流形代表了数据的高密度区域。它们将数值步骤分解为梯度部分和非线性转移部分。转移部分确保即使轨迹是弯曲的，结果状态 x_t 仍被约束在数据流形上 [Liu et al., 2022]。此外，阐明扩散模型（EDM）使用了二阶 Heun 求解器 [Karras et al., 2022]。通过将网络输入、输出和损失缩放到单位方差，它稳定了训练动态并改进了分数估计。

4.4.2 轨迹修正

如前所述，流匹配旨在学习一个向量场 $v_\theta(\mathbf{x}(t), t)$，该向量场能够诱导一个概率流，从而将简单的先验分布 p_T 转换为复杂的数据分布 p_0。其核心目标是最小化学到的速度场与将样本沿选定路径传输的真实条件速度之间的差异（公式 101）。然而，问题源于数据分布 p_0 与噪声分布 p_T 之间的耦合。

在初始训练阶段，通常称为 1-修正流，数据点 $\mathbf{x}(0)$ 和噪声点 $\mathbf{x}(T)$ 从其各自的分布中随机配对。这种随机配对导致一种现象：高维潜在空间中的轨迹经常相交。当多条直线轨迹在时空中的同一点 $\mathbf{x}(t)$ 相交时，学到的边际向量场 $v_\theta(\mathbf{x}(t), t)$（即这些不同相交速度的期望）会变得弯曲。因此，在积分常微分方程时实际遵循的路径不再是直的，这导致在数值求解器中采用大步长时会产生更多的离散化误差 [Zhang et al., 2025b; Yang et al., 2025a]。在修正流框架中，可以通过减少条件路径之间的干扰来改善噪声分布与数据分布之间的耦合。首先，使用随机配对 $(\mathbf{x}(0), \mathbf{x}(T))$ 训练一个 1-修正流 v_θ^1。然后，使用训练好的模型 v_θ^1 生成一个新的合成数据集。具体来说，对于一组噪声样本 $\{\mathbf{x}_i(T)\}$，求解常微分方程 $\frac{d\mathbf{x}(t)}{dt} = -v_\theta^1(\mathbf{x}, t)$ 以获得对应的生成数据样本 $\{\hat{\mathbf{x}}_i(0)\}$。这样就创建了一组确定性的、重新连接的对 $(\hat{\mathbf{x}}_i(0), \mathbf{x}_i(T))$。最后，使用这些对齐的对来训练一个新模型 v_θ^2（称为 2-修正流）。由于 $\hat{\mathbf{x}}_i(0)$ 是通过第一个模型的动力学从 $\mathbf{x}_i(T)$ 生成的，这些潜在空间中的对是高度对齐的。因此，得到的向量场明显更加线性。

对直线轨迹的探索根植于最优传输原理和 Wasserstein 距离的最小化。该领域的一个重要结果是，对于从高斯分布到两个高斯混合分布的修正流，只需几次修正就足以实现直线流。这一见解支撑了 InstaFlow 和 Stable Diffusion 等模型的经验成功，这些模型利用修正流的直线化特性来减少函数评估次数，同时保持样本质量 [Esser et al., 2024]。这些商业化模型的成功也表明，采用修正流的扩散模型可以很好地扩展用于高质量图像生成。

轨迹的直线性还与学到的向量场的 Lipschitz 常数有关。如果 1-修正流是 L-Lipschitz 的，则可以防止附近的噪声样本映射到任意远的数据点。然而，在实践中，L 可能很大，导致在生成的早期阶段传输路径出现显著弯曲。这种弯曲会降低一步欧拉采样的精度，可能需要进一步的修正或更先进的采样调度。

轨迹修正的实际实现也涉及架构改进。研究人员已用复杂的多模态扩散 Transformer（MMDiT）取代了简单的 U-Net 架构。例如，Stable Diffusion 3 引入了 MMDiT 架构 [Esser et al., 2024]，而 Flux.1 进一步将修正流 Transformer 架构扩展到 120 亿参数 [Greenberg, 2025]。这表明扩散模型可以扩展到非常大的规模，同时保持高采样效率。

为了进一步提高效率和稳定性，也可以将分治策略应用于修正流。例如，分段修正流（PeR-Flow）将生成轨迹划分为几个离散的时间窗口 [Yan et al., 2024]。它不是拉直全局流，而是单独拉直每个区间内的片段，这种局部化降低了模拟训练轨迹的计算成本，并最小化了数值误差。此外，研究人员通过先进的引导技术解决了条件生成的稳定性问题。例如，Rectified-CFG++ 采用了一种自适应预测器-校正器方案 [Saini et al., 2025]。每个生成步骤首先执行条件更新，将样本锚定在学到的传输路径附近，然后基于条件速度与无条件速度之间的差异应用流形感知校正。这确保了采样轨迹保持在数据流形的一个稳定管状邻域内。

4.4.3 一致性模型

一致性建模代表了一种从迭代积分转向直接映射范式的方法转变。不同于沿轨迹路径逐步求解，一致性模型旨在学习一个函数，该函数能够将概率流 ODE 轨迹上任意时间步的任意点直接映射至其原点 [Song et al., 2023]。这种自一致性特性确保同一条路径上的所有点都会生成相同的输出图像，因此我们可以通过单步/少步生成获得高质量结果。

为实现这一目标，一致性模型通常通过两种机制进行训练：一致性训练与一致性蒸馏。虽然一致性训练支持从零开始训练，但实践证明一致性蒸馏在大规模应用中更具实用性——它将预训练标准扩散模型（教师模型）的知识提炼至一致性模型（学生模型）。蒸馏损失函数通过最小化学生模型在当前时间步 t 的预测与下一时间步预测之间的差异，将教师模型的多步轨迹压缩为单步跳跃。为进一步提升生成质量（特别是单步采样场景），研究者引入了对抗损失函数来确保预测样本与真实数据分布不可区分 [Sauer et al., 2024]。

该范式的扩展版本是潜在一致性模型（LCM），它将一致性蒸馏应用于 Stable Diffusion 等高分辨率图像合成模型的潜在空间 [Luo et al., 2023a]。通过提炼潜在 ODE 的概率流，LCM 在保持母模型高分辨率能力的同时显著降低了推理延迟。此外，为缓解重训练大型骨干网络的计算开销，LCM-LoRA 提出了一种参数高效的微调策略。该方法训练轻量级的低秩自适应（LoRA）模块作为通用加速器，使得各类微调后的扩散模型无需修改预训练参数即可获得一致性特性 [Luo et al., 2023b]。

尽管早期 LCM 实现了显著的加速效果，但随着采样步数增加，其生成质量往往会出现下降。针对此问题，轨迹一致性蒸馏（TCD）受指数积分器启发，在半线性框架中重构一致性函数以实现对任意中间步骤的映射 [Zheng et al., 2024]。而分阶段一致性模型（PCM）则将轨迹划分为多个子轨迹，从而在避免随机误差累积的前提下实现确定性多步采样 [Wang et al., 2024]。近期如简化一致性模型（sCM）等连续时间模型还引入了 TrigFlow 公式来稳定训练过程 [Lu and Song, 2025]。

需注意的是，虽然一致性模型针对单步生成进行了优化，但它们同样支持多步一致性采样。通过执行 2 至 4 步采样并对中间输出重新编码，模型可以迭代修正离散化误差。这为推理速度与样本保真度之间提供了灵活的权衡空间。

5. 语言中的扩散模型

我们已经看到，图像生成可以被视为状态沿着微分方程轨迹的演化过程。由此产生的扩散模型为描述先验分布与数据分布之间的双向转换提供了强大工具。本节我们将这种方法扩展到词元序列的生成——这是现代**自然语言处理**（NLP）领域的核心问题。

关键挑战在于词元是携带语言意义的离散变量。具体而言，与图像的连续空间不同，两个离散词元之间不存在有意义的中间状态。此外，语言的类别性与序列性意味着即使词元身份或顺序

的微小变化，也可能彻底改变语言含义。因此，我们需要通过改进现有方法与算法，使扩散模型适应词元序列建模。本节首先介绍自回归与非自回归生成范式，随后探讨语言扩散建模的两种方法：嵌入空间扩散与离散扩散。前者基于连续时间扩散模型，在词元的嵌入空间中操作；后者直接在离散词元空间定义生成过程，通常利用词元掩码与预测机制。此外，我们还将讨论扩散 Transformer 架构与流匹配在词元序列生成中的应用，以及推理与生成控制的相关问题。

扩散建模是一个涵盖广泛的主题。在此，我们尝试从微分方程的角度进行讨论，从而将关键概念与前文内容相联系。同时也会介绍那些最初并非由微分方程驱动、但为保持讨论完整性而纳入的方法。

5.1 令牌序列生成

我们首先介绍**自回归（AR）生成**和**非自回归（NAR）生成**的概念，然后证明扩散模型可以被视为一个 NAR 生成问题。

5.1.1 自回归生成与非自回归生成

令牌序列生成（简称**序列生成**）是自然语言处理中的基础任务，也是**大语言模型**（LLMs）的基石 [Radford et al., 2018]。主流方法是自回归（AR）生成[11]。自然语言处理中通常采用从左到右的生成方式，即根据已生成的所有前序令牌逐个预测后续令牌。形式化地，设 $\mathbf{x} = \{x_1, ..., x_n\}$ 为令牌序列，其概率通过概率链式法则定义为：<

$$\begin{aligned} \Pr(\mathbf{x}) &= \Pr(x_1, ..., x_n) \\ &= \prod_{i=1}^{n} \Pr(x_i | x_{<i}). \end{aligned} \tag{103}$$

>

其中 x_i 表示当前步生成的令牌，$x_{<i}$ 表示所有已生成的前序令牌。$\Pr(x_i|x_{<i})$ 是下一令牌的条件概率，体现了从左到右生成的本质。实现 $\Pr(x_i|x_{<i})$ 的方式多样，常用方法是使用神经网络建模该概率，例如 LSTM 和 Transformer 都是神经语言建模中的经典架构。

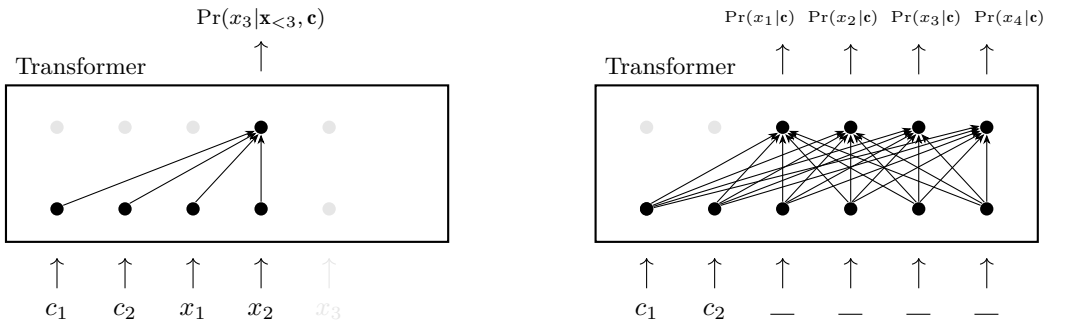
多数文本生成任务需要基于用户提供的上下文信息进行条件生成。我们将这类上下文信息记为 $\mathbf{c}$，可广义视为变量序列 $\mathbf{c} = \{c_1, ..., c_m\}$。此时生成任务转化为建模给定 $\mathbf{c}$ 时目标序列 $\mathbf{x}$ 的概率：

$$\Pr(\mathbf{x}|\mathbf{c}) = \prod_{i=1}^{n} \Pr(x_i | x_{<i}, \mathbf{c}). \tag{104}$$

此处 $\mathbf{c}$ 可以是离散令牌序列（如提示词）或实值向量序列（如神经网络生成的表示）。通常 $\mathbf{c}$ 可根据具体任务定义，例如在翻译中作为源语句，在问答中作为提示词，或作为外部记忆中的历史表示。

自回归生成的核心优势在于其简洁性，由此衍生出多项优良特性：既符合人类语言产出的认知过程，也适合对话、叙事等任务；还能自然处理变长序列——模型持续生成直至输出序列结束符（EOS），而非填充固定长度模板。但自回归方式长期受学界诟病的缺点包括：生成速度慢，由

11. 统计学中，回归指估计因变量（Y）与自变量（X）的关系。前缀 *auto* 源自希腊语" 自我"，因此自回归表示用变量自身的滞后值作为自变量进行回归分析。即不通过 X 预测 Y，而是用 $Y_{\text{昨日}}$ 预测 $Y_{\text{今日}}$



(a) Autoregressive Generation (Single Step)

(b) Non-autoregressive Generation (Single Step)

图 16: 自回归生成与非自回归生成对比。$\mathbf{c} = c_1, c_2$ 表示用户提供的上下文，$\mathbf{x} = x_1, x_2, x_3, x_4$ 表示目标序列。我们的目标是对给定 $\mathbf{c}$ 条件下 $\mathbf{x}$ 的概率进行建模。(a) 在自回归生成中，词元按序预测，x_i 的预测依赖于历史信息 $\mathbf{x}_{<i}$ 和 $\mathbf{c}$。(b) 在非自回归生成中，序列中的所有词元并行预测。图 (b) 中的虚线表示一个由 4 个掩码词元组成的模板（给定目标长度时）。

于第 i 步依赖第 $i-1$ 步，无法并行生成，导致 100 个令牌的句子需顺序执行 100 次前向计算，实时应用计算成本高；上下文视野受限，预测 x_i 时仅能观测左侧上下文 $x_{<i}$，无法前瞻右侧语义。

非自回归（NAR）生成通过打破序列依赖链提供替代方案 [Gu et al., 2018]。与逐个生成词元不同，非自回归模型并行地生成序列中的所有词元。从数学上讲，这种方法依赖于目标词元在给定输入条件下相互独立的假设。此时，序列 $\mathbf{x}$ 的概率可分解为：

$$\Pr(\mathbf{x}|\mathbf{c}) \quad = \quad \prod_{i=1}^{n} \Pr(x_i|\mathbf{c}) \tag{105}$$

在这种形式中，词元 x_i 的预测不依赖于其他目标词元 x_j $(j \neq i)$。因此，非自回归生成具有高效的推理性能。通过解耦词元之间的依赖关系，模型可以在单次前向传播中预测整个序列，从而将推理步骤从 $O(n)$ 减少到 $O(1)$。这种并行性使得现代 GPU 能够得到更高效的利用，因此对于延迟要求苛刻的实时应用而言，非自回归模型具有很大吸引力。此外，非自回归模型在生成过程中可以充分利用过去和未来位置的信息。图 16 展示了自回归生成与非自回归生成之间的区别。

然而需要指出的是，条件独立性假设也存在局限性。例如，在自然语言中，一个句子往往存在多种有效的补全方式。非自回归模型由于无法知晓其他位置已选择了哪些词元，可能会混合不同的有效模式，从而导致所谓的多峰性问题[12]。

5.1.2 扩散建模作为非自回归生成

尽管存在上述限制，但非自回归生成的并行特性与扩散模型完美契合，扩散模型通常同时对全部数据维度进行操作。因此，自然语言处理中的大多数扩散模型都基于非自回归生成范式。为了理解这种联系，将非自回归生成的概念推广到单步预测之外是有帮助的。让我们将生成过程重新想象为一个随时间 t 演化的动力系统。我们将 $\mathbf{s}(t)$ 定义为系统在时间 t 的中间状态。这个抽象

12. 此问题可能导致重复、遗漏或语法不连贯，例如当有效实体为“New York”或“London”时，模型却可能在位置 i 选择“New”，在位置 $i+1$ 选择“London”。

状态 $\mathbf{s}(t)$ 作为一个统一的表示：它既可以是一系列离散变量（例如，被破坏的标记），也可以是一系列连续向量（例如，标记嵌入）。在生成过程中，系统从 $t = T$ 到 $t = 0$ 在时间上反向演化。该系统的状态定义如下

- 起始点 $\mathbf{s}(T)$。这表示从先验分布（例如高斯噪声向量或完全随机的标记）中抽取的样本。它不包含任何语义信息。
- 中间状态 $\mathbf{s}(t)$（当 $0 < t < T$ 时）。这些状态表示不断演化的潜在变量。它们可视为目标数据的带噪近似。
- 终止点 $\mathbf{s}(0)$。状态 $\mathbf{s}(0)$ 表示完全去噪后的信号。最终，通过确定性或随机性映射（例如取整或 argmax 操作）将 $\mathbf{s}(0)$ 投影回有效的标记序列 $\mathbf{x} = \{x_1, \ldots, x_n\}$。

在这个过程中，我们需要将条件 $\mathbf{c}$ 合并到每个状态 $\mathbf{s}(t)$ 中，以形成联合状态 $(\mathbf{s}(t), \mathbf{c})$。因此，演化可以描述为

$$(\mathbf{s}(T), \mathbf{c}) \to \cdots \to (\mathbf{s}(t), \mathbf{c}) \to \cdots \to (\mathbf{s}(0), \mathbf{c}) \to \mathbf{x}. \tag{106}$$

最终的映射 $(\mathbf{s}(0), \mathbf{c}) \to \mathbf{x}$ 通常通过使用一个单独的系统来执行（例如，将嵌入映射到标记）。步骤 $(\mathbf{s}(T), \mathbf{c}) \to \cdots \to (\mathbf{s}(t), \mathbf{c}) \to \cdots \to (\mathbf{s}(0), \mathbf{c})$ 可以被视为一个动力系统的演化，就像标准扩散模型中那样。这个过程如图 17 所示。

在本节的剩余部分，我们将使用扩散模型作为实现非自回归生成的工具。我们根据它们如何定义动力系统 $\mathbf{s}(t)$ 的状态空间对这些方法进行分类。我们首先介绍嵌入空间扩散，它将标记映射到连续向量以应用连续高斯扩散。然后我们讨论离散扩散，它直接在离散标记上定义破坏和去噪过程。

5.2 连续空间中的扩散：嵌入空间扩散

在本小节中，我们介绍嵌入空间扩散框架，该框架将连续扩散模型适配于词元序列生成。

5.2.1 通用框架

将扩散模型应用于文本的一种直观方法是，将离散词元映射到连续嵌入空间 $\mathbb{R}^d$ 中，从而使扩散和生成过程可以在连续域中进行。几项重要工作都基于这种嵌入空间扩散框架 [Li et al., 2022b; Strudel et al., 2022; Gong et al., 2023]。在这些方法中，第一步是将序列中的每个词元转换为一个 d 维实值向量（称为词元嵌入）。形式上，令 V 为一个大小为 $|V|$ 的词表。每个词元 $x \in V$ 通常表示为一个独热向量，并投影为一个稠密向量。然后，一个词元序列 $\mathbf{x} = \{x_1, \ldots, x_n\}$ 可以通过一个嵌入函数 $\text{Embed}(\cdot)$ 映射为一个连续向量序列，如下所示：

$$\begin{aligned} \text{Embed}(\mathbf{x}) &= [\mathbf{e}_1, \ldots, \mathbf{e}_n], \\ &= \mathbf{e} \end{aligned} \tag{107}$$

其中 $\mathbf{e}_i \in \mathbb{R}^d$ 表示位置 i 处的嵌入，$\mathbf{e} \in \mathbb{R}^{d \times n}$ 表示嵌入序列（堆叠为矩阵）。同样，我们可以将上下文词元序列 $\mathbf{c}$ 转换为嵌入序列 $\mathbf{e}_c \in \mathbb{R}^{d \times n_c}$。

定义 $\text{Embed}(\cdot)$ 的一种简单方法是使用一个静态嵌入矩阵 $\mathbf{E} \in \mathbb{R}^{d \times |V|}$ [Mikolov et al., 2013a;b]。这种方法虽然简单，但将词元视为独立单元，从而忽略了它们在序列中的上下文依赖关系。或者，

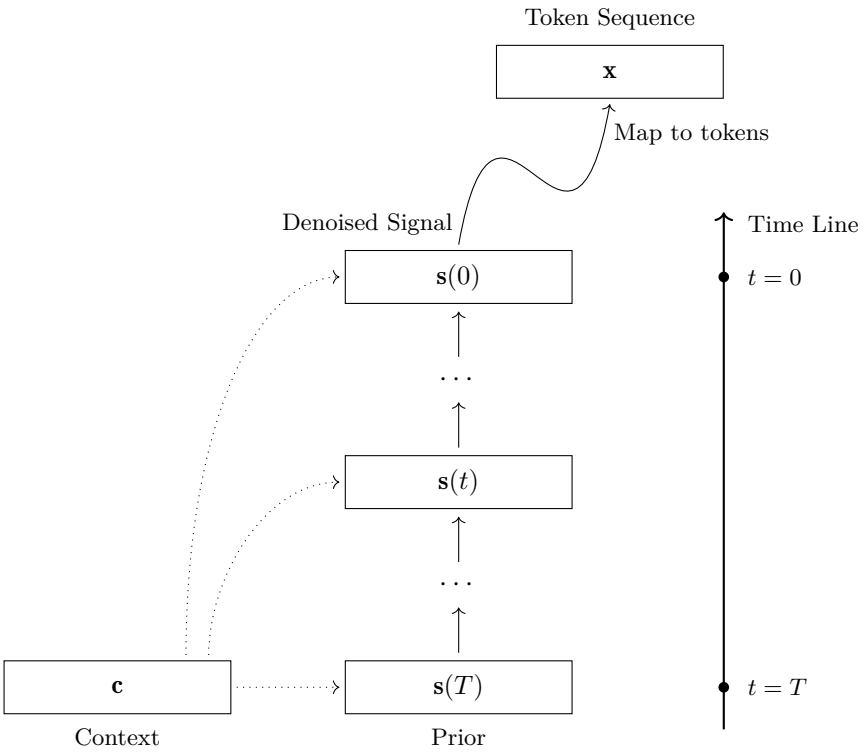


图 17: 基于扩散模型的非自回归生成示意图。生成过程被建模为状态 $\mathbf{s}(t)$ 随时间 t 的演化过程。其中 $\mathbf{s}(T)$ 表示起始点（先验分布），$\mathbf{s}(0)$ 表示终点（完全去噪信号）。上下文信息 $\mathbf{c}$ 在生成过程中被注入状态变量。最终输出是从 $\mathbf{s}(0)$ 映射得到的词元序列。

可以将整个词元序列映射到一个上下文化表示序列中。例如，可以使用预训练的大语言模型来编码词元及其周围的上下文 [Devlin et al., 2019; Brown et al., 2020]。请注意，这种变换不限于词元与向量之间的一一对应关系，也不受限于固定的序列长度 n。相反，我们可以使用更高层次的抽象将词元序列映射到一个不同长度 n' 的潜在连续序列（其中 n' 不必等于 n）。例如，通过使用变分自编码器，可以将离散序列压缩为更稠密、语义更丰富的潜在表示。在这种潜在扩散范式 [Rombach et al., 2022] 中，扩散过程作用于这些压缩后的潜在变量，而不是原始的词元嵌入。

一旦词元被表示为嵌入，生成问题就可以用标准方式通过扩散模型来形式化。形式上，令 $\mathbf{e}_c$ 为上下文嵌入序列。我们定义 $\bar{\mathbf{s}}(t)$ 为结合了 $\mathbf{e}_c$ 和 $\mathbf{s}(t)$ 的增广状态：

$$\bar{\mathbf{s}}(t) \quad = \quad [\mathbf{e}_c, \mathbf{s}(t)]. \tag{108}$$

我们将 $\mathbf{e}_c$ 和 $\mathbf{e}$ 的拼接视为连续扩散过程的初始状态 $\bar{\mathbf{s}}(0) = [\mathbf{e}_c, \mathbf{e}] \in \mathbb{R}^{d\times(n_c+n)}$。然后，我们定义一个前向过程，在时间 $t \in [0, T]$ 内逐渐向目标嵌入添加高斯噪声，同时保持上下文嵌入固定。它将有意义的词元嵌入转换为标准高斯噪声 $\mathcal{N}(\mathbf{0}, \mathbf{I})$。

如前所述，有几种模型可供选择。例如，我们可以用一个随机微分方程来描述扩散过程：

$$d\bar{\mathbf{s}}(t) \quad = \quad f(\bar{\mathbf{s}}(t), t)dt + g(t)d\mathbf{w}, \tag{109}$$

其中 $f(\bar{\mathbf{s}}(t), t)$ 是漂移系数，$g(t)$ 是扩散系数，$\mathbf{w}$ 是一个标准维纳过程。生成过程对应于反向时间随机微分方程，该方程从目标序列部分为高斯噪声的状态 $\bar{\mathbf{s}}(T)$ 重建 $\bar{\mathbf{s}}(0)$：

$$d\bar{\mathbf{s}}(t) = [f(\bar{\mathbf{s}}(t), t) - g(t)^2 \nabla_{\bar{\mathbf{s}}(t)} \log p_t(\bar{\mathbf{s}}(t))]dt + g(t)d\bar{\mathbf{w}}, \tag{110}$$

其中 $\bar{\mathbf{w}}$ 是反向时间维纳过程，$\nabla_{\bar{\mathbf{s}}(t)} \log p_t(\bar{\mathbf{s}}(t))$ 是得分函数。或者，可以使用流匹配来学习一个向量场，该向量场将分布从高斯噪声向前推送到数据分布。这些方法的详细讨论在此省略。读者可以参考第 4 节，以获取关于各种扩散模型的更多细节。

请注意，由于上下文 $\mathbf{c}$ 及其嵌入 $\mathbf{e}_c$ 在整个过程中保持不变，我们可以简单地使用 $\mathbf{s}(t)$ 而不是 $\bar{\mathbf{s}}(t)$ 来描述扩散和生成过程。因此，方程 (109) 和 (110) 可以等价地重写为：

$$d\mathbf{s}(t) = f(\mathbf{s}(t), t)dt + g(t)d\mathbf{w}, \tag{111}$$

$$d\mathbf{s}(t) = [f(\mathbf{s}(t), t) - g(t)^2 \nabla_{\mathbf{s}(t)} \log p_t(\mathbf{s}(t))]dt + g(t)d\bar{\mathbf{w}}. \tag{112}$$

与计算机视觉模型的一个关键区别在于，在自然语言处理中，我们通常使用 Transformer 作为学习得分函数或向量场的主干架构。给定一个状态 $\bar{\mathbf{s}}(t) = [\mathbf{e}_c, \mathbf{s}(t)] \in \mathbb{R}^{d \times (n_c + n)}$，Transformer 输出一个相同形状的实值矩阵，估计每个位置处的得分或向量场，如图 18 所示。与图像生成（其空间维度通常是固定的）不同，自然语言序列的长度本质上是可变的。因此，我们需要在推理过程中确定目标序列长度 n。此外，尽管扩散和生成过程在连续空间中进行以获得 $\bar{\mathbf{s}}(0)$，但目标是产生离散词元。因此，需要一个额外的模块将生成的连续嵌入解码回离散词元。我们将在下面简要讨论这两个挑战。

5.2.2 长度预测

长度预测是自然语言处理中一个反复出现的问题，已有悠久的研究历史。一种直接的方法是根据上下文明确预测目标序列的长度。例如，可以通过将源长度乘以特定比率来简单地推导目标长度。这一概念可以追溯到 ** 统计机器翻译 **（SMT）的早期，当时使用繁衍模型来确定与单个源词对应的目标词数量 [Brown et al., 1993]。在非自回归生成模型的发展过程中也采用了类似的机制 [Xiao et al., 2023]。在早期的非自回归生成模型（通常是编码器-解码器架构）中，编码器中集成了一个繁衍预测器。该预测器估计每个编码器隐藏状态应被复制到解码器的次数，从而在解码之前规划目标序列长度。在这类方法中，长度预测被构建为对概率 $\Pr(n|\mathbf{c})$ 进行建模。预测器通常是一个神经网络，例如 ** 多层感知机 **（MLP），其训练目标是最大化给定上下文条件下真实长度的似然。在推理过程中，通过选择概率最高的长度 n^* 来确定目标长度。

显式长度预测方法简单，但常常导致重复和幻觉，因为模型会试图用无意义或冗余的令牌来填充预分配窗口中未使用的容量。相反，当前大多数扩散语言模型采用隐式长度确定方法。这种方法与自回归生成的行为一致，允许模型自行发出终止信号。在实践中，生成过程从一个足够长的窗口（例如，最大序列长度）开始。一旦该过程完成，并且连续嵌入已被解码为离散令牌，系统会识别第一个终止标记（如 EOS 令牌）的出现位置。该位置被视为句子的逻辑结尾，其后的所有令牌都将被丢弃。

最近的研究也通过引入动态长度适应机制，解决了预分配最大长度窗口带来的计算效率低下的问题 [Li et al., 2025; Yang et al., 2025b]。这些方法不依赖于仅通过 EOS 令牌进行事后截断，而是将长度调制直接集成到扩散和生成过程中。通过离散的插入和删除操作或灵活的去噪调度，它们可以动态调整序列长度。

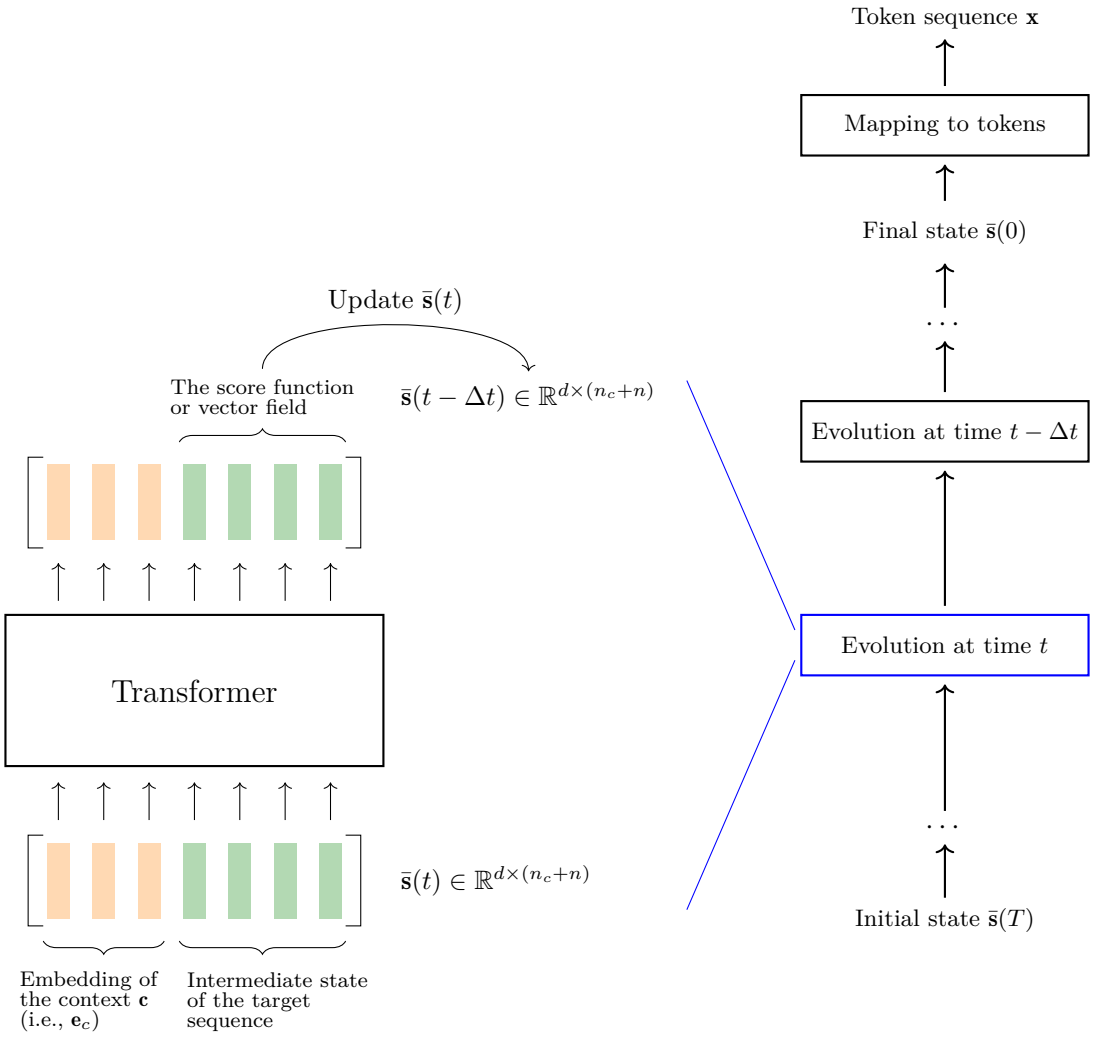


图 18: 基于 Transformer 的生成过程示意图。生成（即反向扩散）过程从目标序列的标准高斯噪声初始化状态 $\bar{\mathbf{s}}(T)$ 开始。在每个时间步 t，状态 $\bar{\mathbf{s}}(t)$ 表示上下文嵌入与含噪目标嵌入的拼接。Transformer 模型基于 $\bar{\mathbf{s}}(t)$ 估计得分函数（或向量场），随后用于将状态更新至 $\bar{\mathbf{s}}(t-\Delta t)$。该迭代过程持续进行直至达到干净状态 $\bar{\mathbf{s}}(0)$，最终被解码为离散标记。注意上下文嵌入 $\mathbf{e}_c$ 在更新过程中保持固定。

5.2.3 舍入

如上所述，生成过程在 $t=0$ 时终止，产生一个连续向量序列 $\mathbf{s}(0)$。我们将 $\bar{\mathbf{s}}(0)$ 内生成的嵌入序列表示为 $\hat{\mathbf{e}} = [\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_n]$，其中每个 $\hat{\mathbf{e}}_i \in \mathbb{R}^d$。由于我们的目标是生成文本，必须将这些连续向量映射回词汇表 V 中的离散词元。此操作通常被称为 **舍入**或解码。

最直接的舍入方法是 **最近邻搜索**。假设我们使用预训练的嵌入矩阵 $\mathbf{E} \in \mathbb{R}^{d\times|V|}$ 来表示词元嵌入，其中每一列对应一个词元 $v \in V$ 的嵌入。那么，我们可以选择在特定距离度量下，其嵌入与生成的向量 $\hat{\mathbf{e}}_i$ 最接近的词元。由于扩散模型通常使用 MSE 损失进行训练，欧几里得距离是一

个自然的选择

$$x_i \quad = \quad \arg\min_{v \in V} \|\hat{\mathbf{e}}_i - \mathbf{e}_v\|_2^2, \tag{113}$$

其中 $\mathbf{e}_v$ 表示 $\mathbf{E}$ 中对应词元 v 的列。或者，也可以采用可训练的投影层后接 softmax 函数，类似于标准自回归语言模型中的输出层。在这种情况下，词元 x_i 的概率由 Softmax 函数给出。

相比之下，对于基于潜在或上下文嵌入的模型（例如使用 VAE 或 LLM 表示的模型），简单的最近邻搜索不适用，因为生成的潜在变量并不直接对应静态的词元嵌入。在这些方法中，需要一个可学习的解码器将连续的潜在序列映射回离散的词元空间。解码过程通常涉及一个神经网络，该网络为每个潜在向量预测词汇表上的概率分布，如同自回归生成模型一样。

舍入并非易事，这源于连续扩散潜在空间与词元嵌入的离散性质之间的不匹配。标准的扩散训练目标（如 MSE）鼓励模型预测数据的期望值。在语言的上下文中，如果分布是多模态的（例如，下一个词元可能是“猫”或“狗”），模型可能会预测它们嵌入的平均值：$(\mathbf{e}_{\text{猫}} + \mathbf{e}_{\text{狗}})/2$。这个平均向量位于有效嵌入之间的间隙空间中。简单地将此向量舍入到最近邻，可能会导致一个语义宽泛的词元（例如“动物”），或者如果嵌入空间不是完全平滑的，则可能导致一个完全不相关的词元。这种现象通常表现为生成的向量未能承诺于一个特定的离散模式。

为了缓解这个问题，研究人员引入了辅助目标和架构修改。Li et al. [2022b] 提出了一个可训练的舍入参数，并向训练目标中添加了一个辅助正则化项。该项明确鼓励模型生成接近 $\mathbf{E}$ 中有效嵌入的向量，从而减少推理过程中的舍入误差。Strudel et al. [2022] 引入了自条件的概念，即将去噪词元的估计值投影回嵌入空间，并用作下一步的输入。这有效地将中间状态钳制在更接近有效数据流形的位置，并使最终状态 $\mathbf{s}(0)$ 能够可靠地映射到离散词元。

5.3 词元空间中的扩散：离散扩散

虽然嵌入空间中的扩散能很好地融入标准扩散模型的框架，但它引入了将连续向量舍入回离散词元的非平凡挑战。为了规避这一问题，并使生成过程更自然地与语言的离散特性对齐，越来越多的研究开始探索离散扩散。在这一范式中，污染过程直接在离散词汇空间 V 上进行操作，从而消除了对嵌入投影和舍入的需求。

5.3.1 CTMC 视角

为了直接在离散词汇表 V 上建模扩散和生成过程，我们摒弃了连续空间中常用的高斯噪声和随机微分方程（SDE）。在离散状态空间中，过程在连续时间内演化，但仅通过瞬时跳跃改变状态：它保持恒定一段随机停留时间，然后跳转到一个新状态。自然描述这种概率跳跃的数学框架是连续时间马尔可夫链（CTMC）。正如 SDE 框架将扩散视为步长趋近于零时随机游走的极限，CTMC 框架将离散扩散视为时间步数趋近于无穷时离散时间马尔可夫链的极限。

由于此处的 CTMC 在离散的 token 上运行，我们使用 $\mathbf{x}(t)$ 而不是 $\mathbf{s}(t)$ 来表示代表 token 序列的状态。令 $\{\mathbf{x}(t)\}_{t \geq 0}$ 为一个随机过程，其中 $\mathbf{x}(t)$ 是时间 t 时由 n 个 token 组成的序列。令 $\mathbf{x}$ 和 $\mathbf{y}$ 表示离散空间 V^n 中的特定状态（即 token 序列）。概率分布 $p_t(\mathbf{x}) = \Pr(\mathbf{x}(t) = \mathbf{x})$ 的演化由 Kolmogorov 前向方程支配

$$\frac{dp_t(\mathbf{x})}{dt} \quad = \quad \sum_{\mathbf{y} \in V^n} p_t(\mathbf{y}) [\mathbf{Q}^{\text{seq}}(t)]_{\mathbf{yx}}, \tag{114}$$

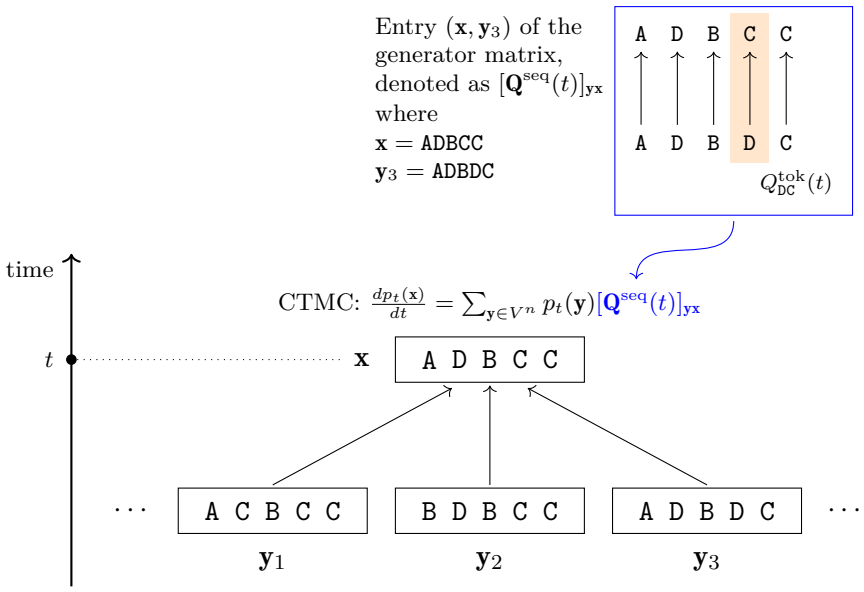


图 19: 连续时间马尔可夫链在标记序列生成中的示意图。该连续时间马尔可夫链在时间 t 的动力学方程为 $\frac{dp_t(\mathbf{x})}{dt} = \sum_{\mathbf{y}\in V^n} p_t(\mathbf{y})[\mathbf{Q}^{\text{seq}}(t)]_{\mathbf{yx}}$。该方程表明，观测到目标序列 $\mathbf{x}$（例如 ADBCC）的概率变化率，是所有可能源序列（例如 $\mathbf{y}_1$、$\mathbf{y}_2$、$\mathbf{y}_3$）的概率之和，并以序列级转移速率 $[\mathbf{Q}^{\text{seq}}(t)]_{\mathbf{yx}}$ 加权。全局转移速率由底层的标记级动力学决定。例如，从序列 $\mathbf{y}_3$ 到 $\mathbf{x}$ 的转移涉及单个标记的变化，例如第四个标记从 D 变为 C 的速率 $Q^{\text{tok}}_{\text{DC}}(t)$。请注意，虽然连续时间马尔可夫链在连续时间中建模动力学，但状态转移本身是离散的跳跃，而非连续变化。

其中 $\mathbf{Q}^{\text{seq}}(t)$ 是整个序列的（时间相关的）生成器矩阵。项 $[\mathbf{Q}^{\text{seq}}(t)]_{\mathbf{yx}}$ 表示从状态 $\mathbf{y}$ 到状态 $\mathbf{x}$ 的瞬时转移率，它是大矩阵 $\mathbf{Q}^{\text{seq}}(t)$ 中的一个元素。

然而，状态空间大小 $|V|^n$ 是指数级大的，使得完整的矩阵 $\mathbf{Q}^{\text{seq}}(t)$ 在计算上不可行。为了解决这个问题，我们通常在前向过程中假设 token 位置之间相互独立。在此假设下，转移在每个位置上独立发生。因此，我们可以使用一个更小的单 token 生成器矩阵 $\mathbf{Q}^{\text{tok}}(t) \in \mathbb{R}^{|V|\times|V|}$ 来建模该过程，该矩阵作用于单个 token。因此，$\mathbf{Q}^{\text{seq}}(t)$ 的复杂高维动力学可以分解为 n 个并行的过程，每个过程都由相同的单 token 生成器 $\mathbf{Q}^{\text{tok}}(t)$ 支配。图 19 展示了方程 (114) 的示意图。

需要注意的是，虽然 CTMC 通过方程 (114) 中的前向方程提供了连续时间的视角，但 NLP 中实用的离散扩散模型通常运行在离散的时间网格 $t \in \{0, 1, \ldots, T\}$ 上。在这种情况下，过程通过相邻时间步之间的转移核（或矩阵）来实现，而 CTMC 表达式可以理解为这些离散化动力学的连续时间极限。

使用 CTMC，我们将前向和后向过程描述如下：

- **前向过程**。在连续时间马尔可夫链（CTMCs）中，我们并非通过添加高斯噪声，而是通过随机标记替换来破坏文本。这一过程由预定义的生成器矩阵 $\mathbf{Q}^{\text{tok}}(t)$ 决定。常用的破坏方案包括：
  - **均匀扩散**：标记以均匀速率跳转到词汇表中的任意其他标记。这类似于连续空间中的高斯噪声，后者将数据分布推向标准正态分布。在此情况下，分布会收敛为词汇表 V 上的均匀分布。

  - **掩码扩散（吸收态）**：标记以特定速率转移至特殊的 `[MASK]` 标记且不可逆，这对应于马尔可夫链中的吸收态。

- **反向过程**。随机过程中的一个标准结果表明，如果前向过程是一个具有生成器 $\mathbf{Q}^{\text{tok}}(t)$ 的 CTMC，那么从 T 到 0 反向运行的后向过程也是一个 CTMC [Anderson, 1982]。其生成速率矩阵 $\tilde{\mathbf{Q}}^{\text{tok}}(t)$ 由下式给出：

$$\tilde{Q}^{\text{tok}}_{vu}(t) \quad = \quad Q^{\text{tok}}_{uv}(t)\frac{p_t(u)}{p_t(v)}, \tag{115}$$

  其中 $u, v \in V$ 代表词汇表中的特定 token 值。这里，$\tilde{Q}^{\text{tok}}_{vu}(t)$ 表示反向过程中从值 v 转移到值 u 的速率，$p_t(v)$ 是 token 在时间 t 取值为 v 的边缘概率。

上述方程建立了 CTMC 与离散扩散模型之间的基本联系。它是反向时间 SDE 公式的离散类比，该公式将反向漂移与得分函数 $\nabla \log p_t(\mathbf{x})$ 联系起来。在离散情况下，概率比 $p_t(u)/p_t(v)$ 充当“离散得分”，引导生成过程朝向高概率的 token。

方程 (115) 是时间反转 CTMC 的理论刻画，其中反向速率依赖于真实的边缘分布 $p_t(\cdot)$。在用于文本的离散扩散模型中，我们通常不显式地估计 $p_t(\cdot)$。相反，反向动力学是通过反向后验 $\Pr(\mathbf{x}(t-\Delta t) \mid \mathbf{x}(t), \mathbf{x}(0), \mathbf{c})$ 的贝叶斯分解来构建的。因此，方程 (115) 主要用于提供直觉，说明为什么概率比或后验重加权会引导反向过程朝向高概率的 token。在实践中，我们使用神经网络（通常是 Transformer）对反向动力学进行参数化，该网络预测一个去噪分布，例如 $p_\theta(\mathbf{x}(0)|\mathbf{x}(t), \mathbf{c})$ [13]。

基于 CTMC 的扩散建模中的一个代表性例子是 Austin et al. [2021] 提出的 **结构化去噪扩散概率模型**（D3PM）。虽然早期尝试离散扩散通常依赖于简单的均匀转移概率，但 D3PM 通过结构化转移矩阵将该框架推广到支持任意噪声过程。重要的是，Austin et al. [2021] 证明了破坏过程的选择会显著影响样本质量。他们引入了几种超越标准均匀扩散的结构化转移方案。在模型参数化方面，D3PM 不同于噪声预测范式。由于噪声在离散空间中不是加性的，D3PM 被训练为直接从噪声输入中预测干净的数据分布 $p_\theta(\mathbf{x}(0)|\mathbf{x}(t), \mathbf{c})$。

需要注意的是，由于在训练期间可以获得真实值 $\mathbf{x}(0)$，我们可以直接计算真实的反向后验 $q(\mathbf{x}(t-\Delta t)|\mathbf{x}(t), \mathbf{x}(0), \mathbf{c})$。因此，这个真实后验作为模型转移概率的监督信号。

形式上，这个反向后验对应于前一个时间步的分布。遵循扩散模型中的常见符号，我们用 q 表示前向条件分布。那么，反向后验可以使用贝叶斯规则推导：

$$q(\mathbf{x}(t-\Delta t)|\mathbf{x}(t), \mathbf{x}(0), \mathbf{c}) \quad = \quad \frac{q(\mathbf{x}(t)|\mathbf{x}(t-\Delta t), \mathbf{c})\, q(\mathbf{x}(t-\Delta t)|\mathbf{x}(0), \mathbf{c})}{q(\mathbf{x}(t)|\mathbf{x}(0), \mathbf{c})}. \tag{116}$$

这里，序列上的概率分布分解为独立的 per-token 转移，例如：

$$q(\mathbf{x}(t)|\mathbf{x}(0), \mathbf{c}) \quad = \quad \prod_{i=1}^{n} q(x_i(t)|x_i(0), \mathbf{c}), \tag{117}$$

13. 在大多数设置中，前向破坏过程被选择为独立于 $\mathbf{c}$。我们在符号中保留 $\mathbf{c}$ 是为了匹配条件生成的设定，并允许更一般的情况，即噪声过程可能依赖于 $\mathbf{c}$。

其中每个概率 $q(x_i(t)|x_i(0),\mathbf{c})$ 由单 token 生成器 $\mathbf{Q}^{\text{tok}}(t)$ 决定 [14]。在实践中，对于像均匀扩散和掩码扩散这样的结构化 $\mathbf{Q}^{\text{tok}}(t)$ 矩阵，前向转移概率 $q(x(t)|x(0),\mathbf{c})$ 具有简单的解析闭式解。对于均匀扩散，我们有：

$$q(x(t)|x(0),\mathbf{c}) \quad = \quad \alpha_t \cdot \mathbb{I}(x(t)=x(0)) + (1-\alpha_t)\cdot\frac{1}{|V|}, \tag{118}$$

其中 α_t 是一个依赖于时间的标量调度参数，控制信号保留率。类似地，对于吸收态（掩码）扩散，概率质量被导向一个特殊的 `[MASK]` token。相应的闭式转移概率由下式给出：

$$q(x(t)|x(0),\mathbf{c}) \quad = \quad \begin{cases} \alpha_t, & \text{if } x(t)=x(0), \\ 1-\alpha_t, & \text{if } x(t)=\texttt{[MASK]}, \\ 0, & \text{otherwise.} \end{cases} \tag{119}$$

这些解析解使我们能够绕过前向过程中计算昂贵的矩阵运算。有了这些易于处理的前向概率，后向过程通过将神经网络对原始 token 序列的预测 $p_\theta(\mathbf{x}(0)|\mathbf{x}(t),\mathbf{c})$ 代入上述推导的贝叶斯分解来实现。这产生了一个反向转移分布 $p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t),\mathbf{c})$，我们可以从中采样。图 20 展示了在假设 $\Delta t=1$ 的情况下，D3PM 中均匀扩散的前向和后向过程。

为了训练模型，我们希望最大化观测数据 $\log p_\theta(\mathbf{x}(0)|\mathbf{c})$ 的对数似然。然而，这需要对所有可能的轨迹 $\mathbf{x}(1{:}T)$ 进行边缘化，这是不可行的。通过引入前向过程 $q(\mathbf{x}(1{:}T)|\mathbf{x}(0),\mathbf{c})$ 作为辅助分布，我们可以将对数边缘概率重写为一个期望：

$$\begin{aligned} \log p_\theta(\mathbf{x}(0)|\mathbf{c}) \quad &= \quad \log \sum_{\mathbf{x}(1:T)} q(\mathbf{x}(1{:}T)|\mathbf{x}(0),\mathbf{c}) \frac{p_\theta(\mathbf{x}(0{:}T)|\mathbf{c})}{q(\mathbf{x}(1{:}T)|\mathbf{x}(0),\mathbf{c})} \\ &= \quad \log \mathbb{E}_{q(\mathbf{x}(1:T)|\mathbf{x}(0),\mathbf{c})}\left[\frac{p_\theta(\mathbf{x}(0{:}T)|\mathbf{c})}{q(\mathbf{x}(1{:}T)|\mathbf{x}(0),\mathbf{c})}\right]. \end{aligned} \tag{120}$$

应用 Jensen 不等式，我们将对数移到期望内部，推导出一个可处理的 **证据下界**（ELBO）：

$$\log p_\theta(\mathbf{x}(0)|\mathbf{c}) \quad \geq \quad \mathbb{E}_{q(\mathbf{x}(1:T)|\mathbf{x}(0),\mathbf{c})}\left[\log \frac{p_\theta(\mathbf{x}(0{:}T)|\mathbf{c})}{q(\mathbf{x}(1{:}T)|\mathbf{x}(0),\mathbf{c})}\right]. \tag{121}$$

通过将联合分布 $p_\theta(\mathbf{x}(0{:}T)|\mathbf{c})$ 和前向轨迹 $q(\mathbf{x}(1{:}T)|\mathbf{x}(0),\mathbf{c})$ 展开为各自条件分布的乘积，然后重新排列项，负 ELBO 可以分解为每步 KL 散度项的和（关于离散设置下的完整推导，请参见

14. 准确地说，$q(x_i(t)|x_i(0),\mathbf{c})$ 对应于单 token 转移概率矩阵 $\mathbf{P}^{\text{tok}}(0,t)=\exp\left(\int_0^t \mathbf{Q}^{\text{tok}}(\tau)\,d\tau\right)$ 的 $(x_i(0),x_i(t))$ 元素（假设随时间可交换）。从物理上讲，$\mathbf{P}^{\text{tok}}(0,t)$ 表示在区间 $[0,t]$ 内 token 之间转移的总概率质量。由于马尔可夫性质，该矩阵也可以被视为连续离散时间步上转移矩阵的乘积：如果我们将区间离散化为 $0=t_0<t_1<\cdots<t_k=t$，那么 $\mathbf{P}^{\text{tok}}(0,t)=\mathbf{P}^{\text{tok}}(t_{k-1},t_k)\times\cdots\times\mathbf{P}^{\text{tok}}(t_0,t_1)$。

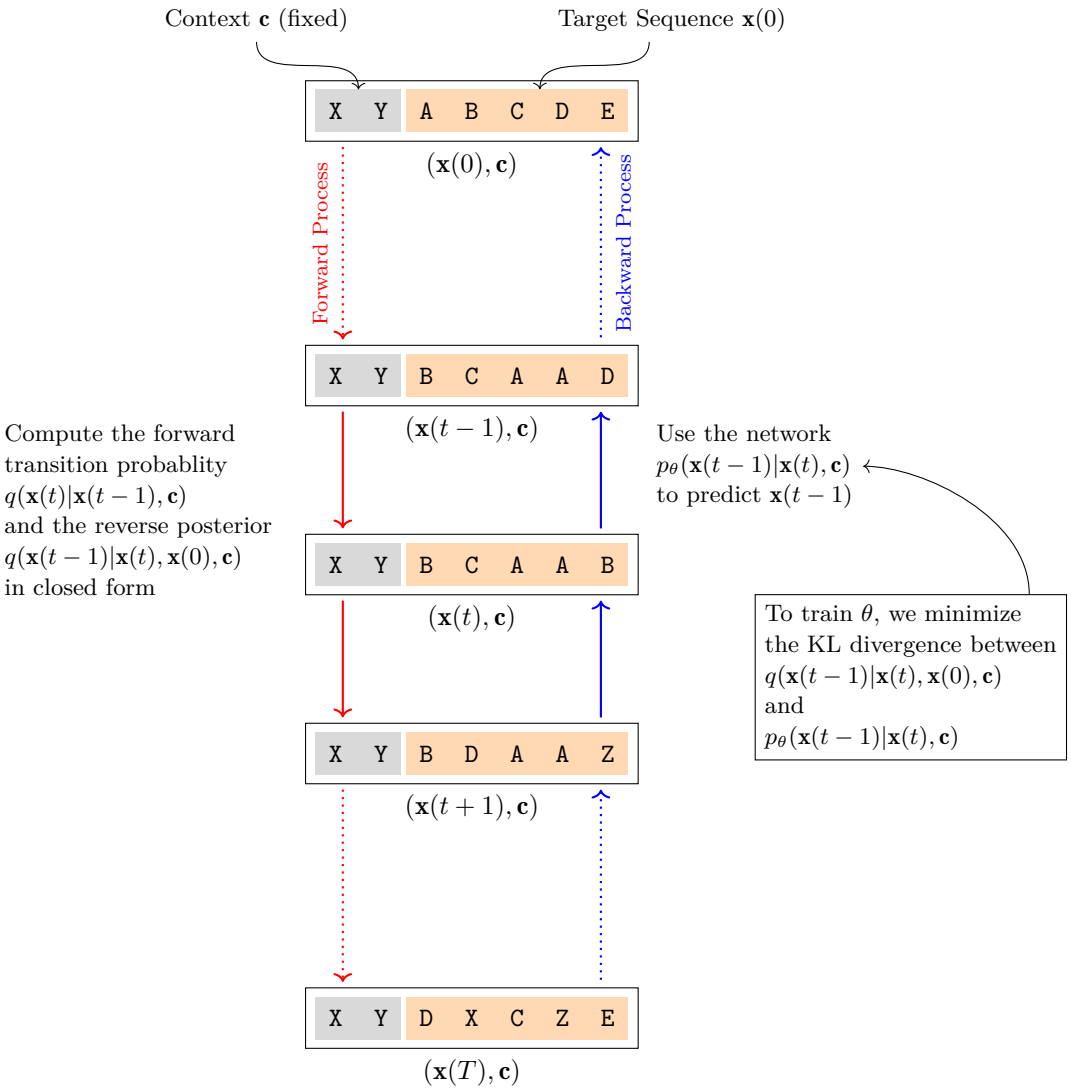


图 20: D3PM 均匀扩散中前向与后向过程的示意图。在前向过程中，状态在每个 $\Delta t = 1$ 的时间步跳转到一个新状态，初始状态为 $(\mathbf{x}(0), \mathbf{c})$，最终状态为 $(\mathbf{x}(T), \mathbf{c})$。在前向步骤中，我们计算前向转移概率 $q(\mathbf{x}(t)|\mathbf{x}(t-1), \mathbf{c})$，然后可用其计算反向后验 $q(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{x}(0), \mathbf{c})$ 以训练后向模型。在后向过程中，模型沿时间反向演化。在每个后向步骤中，我们使用网络 $p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{c})$ 预测词元序列 $\mathbf{x}(t-1)$，然后继续处理 $t-1$。网络的训练目标是，在 q 的期望下，最小化 $q(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{x}(0), \mathbf{c})$ 与 $p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{c})$ 之间的 KL 散度。

[Austin et al., 2021])。模型最小化的最终训练目标由下式给出：

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{x}(1:T)|\mathbf{x}(0),\mathbf{c})}\Bigg[-\log p(\mathbf{x}(T)) + \sum_{t=1}^{T} \text{KL}\Big(q(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{x}(0), \mathbf{c}) \,\|\, p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{c})\Big)\Bigg], \quad (122)$$

在这个目标中，项 $q(\mathbf{x}(t-1)|\mathbf{x}(t),\mathbf{x}(0),\mathbf{c})$ 表示反向转移的真实后验。这里的一个关键优势是，对于结构化的前向过程（如均匀噪声或掩码），这个后验分布可以使用贝叶斯规则高效地以闭式计算。关于第一项，由于先验 $p(\mathbf{x}(T))$ 通常是一个没有可学习参数的固定分布（例如均匀分布或确定性的掩码状态），$-\log p(\mathbf{x}(T))$ 相对于 θ 是常数，可以省略。因此，核心的学习信号驱使模型 p_θ 将其反向转移概率与前向动力学所决定的理想反向路径对齐。

5.3.2 掩码扩散模型

掩码扩散模型（Masked Diffusion Models, MDMs）是离散扩散的一种专门实现，其转移速率矩阵 $\mathbf{Q}^{\text{tok}}(t)$ 被设计为将概率质量移向一个特殊的吸收态，记作 `[MASK]` 标记 [Sahoo et al., 2024; Nie et al., 2025]。这种方法提供了一个独特的实践优势：一旦一个标记在前向过程中转移到掩码状态，它就会停留在那里，从而创建了一条通向完全损坏状态的单向路径。

前向过程逐步将原始序列中的标记替换为 `[MASK]`。它从一个完全观测到的序列开始，逐渐破坏信息，直到整个序列变为仅包含掩码的序列。在实践中，我们将时间离散化，并在离散时间步 $\{0,1,\dots,T\}$ 上实现该过程为一个离散时间马尔可夫链。令 $\mathbf{x}(t)$ 为时间 $t\in\{0,1,\dots,T\}$ 的损坏序列，其中 $\mathbf{x}(0)$ 是干净数据，$\mathbf{x}(T)=\texttt{[MASK]}^n$，$x_i(t)$ 是 $\mathbf{x}(t)$ 的第 i 个标记。在前向过程中，假设各位置之间独立，掩码扩散可以写为每个标记的吸收转移：

$$q(\mathbf{x}(t)|\mathbf{x}(0),\mathbf{c}) \quad = \quad \prod_{i=1}^{n} q(x_i(t)|x_i(0),\mathbf{c}), \tag{123}$$

其中单标记前向核为：

$$q(x_i(t)|x_i(0),\mathbf{c}) \quad = \quad \alpha_t\cdot\mathbb{I}(x_i(t)=x_i(0))+(1-\alpha_t)\cdot\mathbb{I}(x_i(t)=\texttt{[MASK]}). \tag{124}$$

这里 $\alpha_t\in[0,1]$ 是一个单调递减的调度函数，控制着在时间 t 保留多少信号。等价地，我们可以将 $\mathbf{x}(t)$ 视为通过采样一个二元掩码指示向量 $\mathbf{m}_t\in\{0,1\}^n$ 生成，其中：

$$m_{t,i}\sim\text{Bernoulli}(1-\alpha_t), \tag{125}$$

$$x_i(t)=\begin{cases}\texttt{[MASK]}, & m_{t,i}=1,\\ x_i(0), & m_{t,i}=0.\end{cases} \tag{126}$$

因此，扩散轨迹可以解释为重复增加掩码位置的集合，直到最终所有位置都被掩码，如图 21 所示。

后向过程从 $t=T$ 运行到 $t=0$。给定上下文 $\mathbf{c}$ 和一个初始的完全掩码序列 $\mathbf{x}(T)=\texttt{[MASK]}^n$，MDM 迭代地将 `[MASK]` 标记替换为真实标记，直到在 V^n 中获得一个完全实例化的序列。形式上，我们定义一个参数化的逆向马尔可夫链：

$$p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t),\mathbf{c}),\quad t=T,T-1,\dots,1, \tag{127}$$

这里 $p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t),\mathbf{c})$ 是逆向条件分布 $\Pr_\theta(\mathbf{x}(t-1)|\mathbf{x}(t),\mathbf{c})$ 的记法，下标 θ 强调逆向时间动力学是由 θ 参数化的。

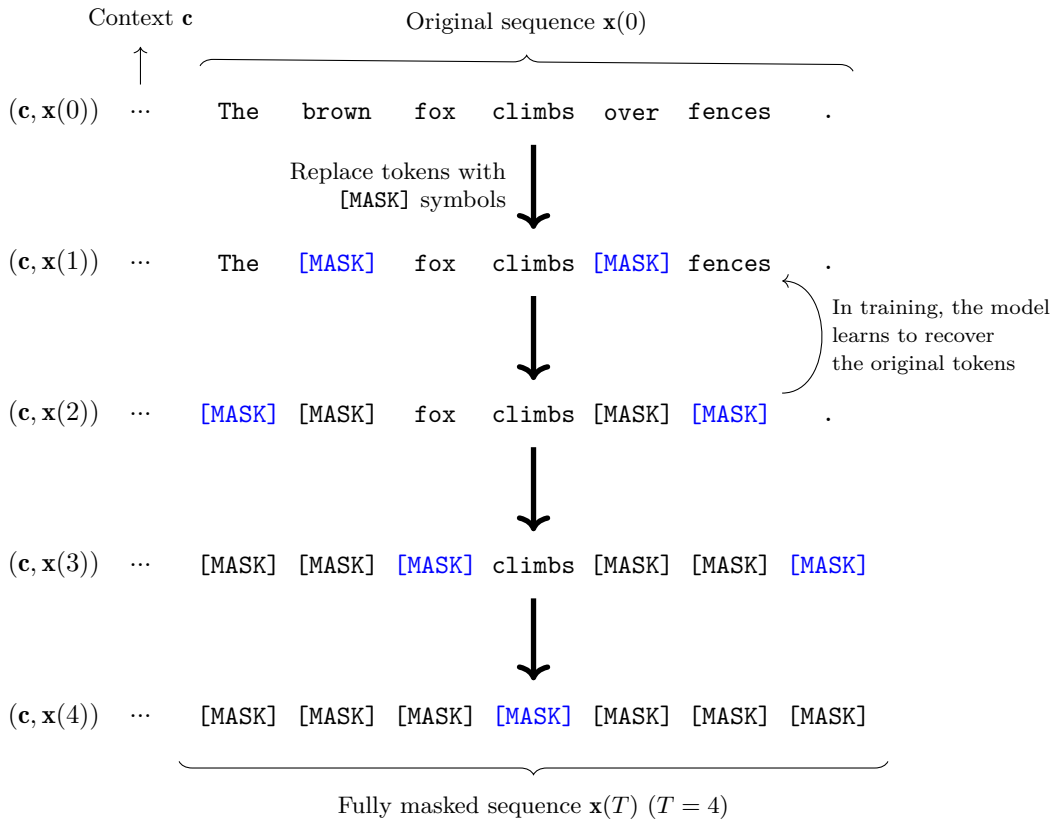


图 21: MDM 前向过程示意图，其中原始标记序列被迭代地转换为完全掩码序列。在此过程中，标记被替换为 [MASK] 符号，直至整个序列被掩码。注意，上下文 $\mathbf{c}$ 在整个过程中始终保持未掩码状态。为训练 MDM，模型学习在固定上下文 $\mathbf{c}$ 和部分损坏序列的条件下，恢复被掩码的标记。

一种简单的参数化方法是让网络预测干净标记的去噪分布：

$$p_\theta(\mathbf{x}(0)|\mathbf{x}(t),\mathbf{c}) = \prod_{i=1}^{n} p_\theta(x_i(0)|\mathbf{x}(t),\mathbf{c}). \tag{128}$$

然后，我们通过仅更新掩码位置来构建逆向转移，同时保持未掩码的标记不变。对于每个位置 i，我们有：

$$p_\theta(x_i(t-1)|\mathbf{x}(t),\mathbf{c}) = \begin{cases} \mathbb{I}(x_i(t-1) = x_i(t)), & \text{if } x_i(t) \neq \texttt{[MASK]}, \\ \pi_{\theta,t,i}(\cdot|\mathbf{x}(t),\mathbf{c}), & \text{if } x_i(t) = \texttt{[MASK]}, \end{cases} \tag{129}$$

其中 $\pi_{\theta,t,i}(\cdot|\mathbf{x}(t),\mathbf{c})$ 是由模型产生的在 V 上的分类分布。一个常见的选择是：

$$\pi_{\theta,t,i}(v|\mathbf{x}(t),\mathbf{c}) = p_\theta(x_i(0) = v|\mathbf{x}(t),\mathbf{c}), \quad v \in V, \tag{130}$$

即，直接从预测的干净标记分布中采样缺失的标记。

给定一个示例数据集 $(\mathbf{x}(0), \mathbf{c}) \sim p_{\text{data}}$，MDMs 通过最大化干净序列的条件似然（或等价地最小化其负对数似然）来学习逆向模型 p_θ。与其他扩散模型类似，我们可以使用 ELBO 来训练 MDMs（另见第 5.3.1 节中的 D3PM）。

尽管上述训练目标看起来有些繁琐，但在实践中，一个更简单且广泛使用的方法是训练模型以恢复掩码位置处的原始标记。我们采样一个时间步 $t \sim U\{1, \dots, T\}$，然后采样一个损坏序列 $\mathbf{x}(t) \sim q(\mathbf{x}(t)|\mathbf{x}(0), \mathbf{c})$。我们定义掩码索引集为：

$$\mathcal{M}(t) \quad = \quad \{i \in \{1, \dots, n\} : x_i(t) = \texttt{[MASK]}\}. \tag{131}$$

损失函数则定义为掩码位置上的条件负对数似然，如下所示：

$$\begin{aligned} & \mathcal{L}_{\text{denoise}}(\theta) \\ = \quad & \mathbb{E}_{(\mathbf{x}(0),\mathbf{c})\sim p_{\text{data}}} \, \mathbb{E}_{t\sim U[T]} \, \mathbb{E}_{\mathbf{x}(t)\sim q(\mathbf{x}(t)|\mathbf{x}(0),\mathbf{c})} \left[- \sum_{i \in \mathcal{M}(t)} \log p_\theta(x_i(0)|\mathbf{x}(t), \mathbf{c}) \right]. \end{aligned} \tag{132}$$

可选地，可以对时间步进行重新加权以强调某些噪声水平：

$$\mathcal{L}_{\text{denoise}}(\theta) \quad = \quad \mathbb{E}\left[\gamma_t \cdot \Big(- \sum_{i \in \mathcal{M}(t)} \log p_\theta(x_i(0)|\mathbf{x}(t), \mathbf{c}) \Big) \right], \tag{133}$$

其中 $\gamma_t \geq 0$ 是一个预定义的调度函数。直观上，类似于课程学习，小的 t 对应轻度损坏（即较容易的去噪），而大的 t 对应重度损坏（即较难的去噪）。权重 γ_t 可以用来平衡这些状态。

训练完成后，我们使用学习到的逆向核 $p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{c})$ 通过从完全掩码状态向后运行马尔可夫链来生成文本。推理过程从以下开始：

$$\mathbf{x}(T) = \texttt{[MASK]}^n, \tag{134}$$

并迭代地采样：

$$\mathbf{x}(t-1) \sim p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t), \mathbf{c}), \quad t = T, T-1, \dots, 1. \tag{135}$$

在方程 (129) 的每标记因式分解下，逆向步骤仅更新掩码位置。对于每个 i，我们有：

$$x_i(t-1) = \begin{cases} x_i(t), & x_i(t) \neq \texttt{[MASK]}, \\ \text{Cat}(\pi_{\theta,t,i}(\cdot|\mathbf{x}(t), \mathbf{c})), & x_i(t) = \texttt{[MASK]}, \end{cases} \tag{136}$$

其中 $\text{Cat}(\cdot)$ 表示从其参数指定的分类分布中采样一个离散标记，即当 $x_i(t) = \texttt{[MASK]}$ 时，$x_i(t-1) \sim \pi_{\theta,t,i}(\cdot|\mathbf{x}(t), \mathbf{c})$。因此，模型在已填充上下文的条件下重复填充缺失标记，直到所有位置都变为未掩码。

上述方法没有指定每步应有多少标记变为未掩码。在实践中，MDMs 通常采用显式的去掩码调度。令 k_t 为从 t 到 $t-1$ 时要新变为未掩码的标记数量，并令 $\mathcal{U}(t) \subseteq \mathcal{M}(t)$ 为要更新的掩码位

置的选定子集，且 $|\mathcal{U}(t)| = k_t$。那么一个常见的实现是：

$$x_i(t-1) = \begin{cases} x_i(t), & i \notin \mathcal{U}(t), \\ \mathrm{Cat}(\pi_{\theta,t,i}(\cdot|\mathbf{x}(t), \mathbf{c})), & i \in \mathcal{U}(t). \end{cases} \tag{137}$$

调度函数 $\{k_t\}_{t=1}^{T}$ 在速度与准确性之间进行权衡：较大的 k_t 导致更少的迭代次数，但要求模型在高不确定性下填充许多标记；而较小的 k_t 产生更多的细化步骤，但增加了计算量。

一个简单的选择是确定性调度，即每步揭示固定的比例（例如，在 t 上呈线性或余弦变化），使得掩码标记的期望数量从 n 减少到 0。另一种常见策略是基于模型置信度选择 $\mathcal{U}(t)$。给定 $\pi_{\theta,t,i}$，我们可以定义一个置信度分数为：

$$s_{t,i} = \max_{v \in V} \ \pi_{\theta,t,i}(v|\mathbf{x}(t), \mathbf{c}), \tag{138}$$

然后我们可以首先对置信度最高的位置进行去掩码。这种“先易后难”的启发式方法倾向于通过用可靠的标记锚定序列来稳定生成，然后再填充更模糊的标记。

5.4 离散扩散的评注

在本小节中，我们讨论在词元空间进行离散扩散时的若干建模考量。

5.4.1 对非吸收式替换扩散的再思考

吸收式的 `[MASK]` 构造具有一种吸引人的信息单调性：前向过程仅移除信息，而反向过程则仅将其填充回来。这种设计与 Transformer 的去噪过程非常契合，并且最近在大规模扩散语言模型的预训练中展现了强大的可扩展性 [Sahoo et al., 2024; Nie et al., 2025]。尽管如此，重新审视非吸收式替换扩散方法（例如均匀扩散）仍然是有价值的，在这些方法中，词元可能在整条轨迹中被反复替换。

首先，均匀替换是一个简单且行为良好的基线。D3PM 中使用的均匀替换核就是一个典型例子。在每一步中，一个词元要么被保留，要么被一个从词汇表中均匀采样的元素所替换。这种策略能得到闭式转移概率，从而保持训练的可处理性。此外，由于这种破坏在原则上是可逆的，反向过程不受不可逆吸收汇的约束。从概念上讲，均匀替换是添加各向同性高斯噪声的离散类比。它驱使分布趋向于一个简单的先验，同时保持对前向边缘分布的分析控制。

其次，在非吸收式替换扩散中，反向更新是密集的。在掩码扩散中，未掩码的词元通常被保留，只有被掩码的位置会被更新。而在非吸收式替换扩散中，每个位置都保持随机性，因此一个自然的反向核可以在每一步更新所有词元。这类似于连续时间扩散模型的结构，其中所有维度在整个轨迹中都被精炼。动态过程可以修正任何组成部分以提高全局一致性，而不是将可见坐标视为固定。

第三，非吸收式替换在语言中可能较为困难。如果词元可以被多次破坏，那么噪声序列 $\mathbf{x}(t)$ 可能包含局部流畅但全局不一致的片段，而反向模型必须同时推断缺失的语义并纠正错误的词元。这与掩码扩散形成对比，在掩码扩散中生成的词元通常是干净的。从连续时间马尔可夫链的视角看，非吸收式替换对应于一个生成器 $\mathbf{Q}^{\mathrm{tok}}(t)$，其所有词元对之间具有非零的非对角速率，因此概率质量在 V 上持续循环，而不是流入单一的汇。随着 t 的增加，信噪比可能比吸收式扩散下降得更快，因为即使是被观测到的词元也是不可靠的。

最近的研究探索了将吸收式掩码与额外的替换噪声相结合的方法，以诱导一种纠正行为。通过这种方式，模型可以检测并修复损坏的词元，而不仅仅是填充被掩码的位置 [Zhang et al., 2025a; Rütte et al., 2025]。这种混合视角表明，非吸收式噪声不仅仅是掩码扩散的竞争者，还可以作为一种互补机制来增强鲁棒性和自校正能力。

一个简单的想法是考虑结构化的替换策略。在自然语言处理中，一种自然的方法是使用依赖于词元或结构感知的转移速率来偏置替换，例如：

$$Q_{uv}^{\text{tok}}(t) \quad = \quad w_{uv}(t), \tag{139}$$

其中 $w_{uv}(t)$ 建模了混淆性，它可以是频率感知的、与子词相关的或基于嵌入相似性的。其动机是使早期噪声在语义上是局部的（易于去噪），而使晚期噪声是全局混合的（接近一个简单的先验）。这类似于连续扩散中的方差保持与方差爆炸设计选择，但现在表示为离散转移结构上的调度。

另一个有趣的方向是在掩码式吸收扩散与均匀替换扩散之间进行插值。例如，可以在生成器层面考虑一种组合：

$$\mathbf{Q}^{\text{tok}}(t) = \lambda_t \cdot \mathbf{Q}_{\text{mask}}^{\text{tok}}(t) + (1 - \lambda_t) \cdot \mathbf{Q}_{\text{unif}}^{\text{tok}}(t), \tag{140}$$

其中模型在某些噪声水平下表现得像掩码扩散，而在其他水平下则保留在未掩码词元之间分配概率质量的能力。最近的研究已经探索了这种组合，以同时获得两种机制的优势（来自吸收式破坏的稳定性和来自替换式破坏的灵活性）。

5.4.2 掩码扩散建模与掩码语言建模

在一般的“损坏-去噪”框架下，掩码扩散模型的训练目标与**掩码语言建模**（MLM）非常接近。例如，在类 BERT 模型中，我们采样一个掩码模式，并训练一个双向 Transformer，使其能够根据部分观察到的上下文来预测掩码位置处的原始词元 [Devlin et al., 2019]。如果我们将式 (126) 解释为以掩码率 $(1 - \alpha_t)$ 采样一个掩码指示符 $\mathbf{m}_t$，那么式 (132) 中的去噪损失本质上就是一个 MLM 交叉熵损失，区别仅在于损坏程度通过 α_t 被时间明确索引，并且网络以扩散步数 t 为条件。因此，主要的概念差异不在于监督信号，而在于去噪器的部署方式。BERT 的 MLM 主要是一种用于表示学习的预训练目标，而掩码扩散模型则将相同的去噪基本操作解释为一个逆时间转移核，并从一个完全掩码的状态开始迭代应用它，以生成一个自然的词元序列。从这个意义上说，现代的掩码扩散模型在架构上通常看起来像 BERT，但它们将 MLM 从一个预训练任务转变为一个序列生成的过程。

掩码扩散模型与 MLM 之间的联系揭示了一个有趣的方向：由于掩码扩散中的前向过程是一种特定的损坏选择，许多为 NLP 预训练开发的加噪策略可以被重新解释为离散扩散的替代前向核。也就是说，除了独立的词元掩码之外，我们还可以改变被损坏的对象（词元与文本片段）、损坏的方式（掩码、替换或置换）以及损坏程度随时间演变的方式（手动设计或学习得到的调度方案）。以下是在设计掩码扩散模型时可以考虑的一些常见加噪策略。

- **非均匀分布的标记替换**。与均匀替换标记不同，可以从频率调整的分布中采样替换项，或从学习到的提议分布中采样。例如 ELECTRA 模型，其中一个小型生成器提出合理的替换项，判别器则被训练来检测它们 [Clark et al., 2019]。从扩散的角度看，这种更困难的非均匀替换可视为注入了比随机标记更具语义混淆性的噪声，这可能会鼓励更强的条件去噪器，并缩小训练时使用的损坏与推理时的不确定性之间的差距。

- **片段损坏**。T5 系列模型将连续的片段替换为单个哨兵标记，并学习重建缺失的内容 [Raffel et al., 2020]。对于扩散式生成，片段损坏是自然的：逆向过程可被解释为逐步将粗糙的缺失区域细化为详细的文本。这与在结构化槽位上迭代去噪而非在独立标记空白处去噪的直觉相符。然而，这种方法的一个挑战在于，将多个标记替换为单个哨兵标记会改变序列长度，这违反了标准扩散模型的固定维度假设。为解决此问题，我们可以采用类似于 SpanBERT 的就地片段掩码 [Joshi et al., 2020]，即连续标记被掩码但序列长度保持不变。或者，可以扩展前向核以显式建模插入和删除操作，从而处理可变长度的轨迹 [Gu et al., 2019; Reid et al., 2022]。

- **标记重排与置换噪声**。BART 引入了带有标记删除和句子置换等损坏的去噪预训练。该方法要求模型不仅学习内容恢复，还要学习重新排序 [Lewis et al., 2020]。将其重新定义为扩散前向核，则意味着存在一类更广泛的离散动态过程，它们同时扰动身份信息和位置信息。这种方法可以导致逆向过程执行联合的“编辑 + 重排序”细化，而非纯粹的填充。

- **对抗性加噪**。另一个方向是利用模型引导的扰动或辅助网络生成更具挑战性的损坏。用于 NLP 的对抗性训练方法（例如，嵌入级扰动）表明，更强、结构化的噪声可以提高鲁棒性 [Zhu et al., 2020]。虽然许多对抗性方法是在连续嵌入空间中定义的，但其核心思想适用于扩散建模。我们可以选择在每个噪声级别上对去噪器施加最大压力的损坏方式，这原则上可以使学习到的逆向动态在迭代采样下更加稳定。

- **学习到的加噪调度**。与其手动固定调度 α_t，不如尝试学习在每个步骤中应用多少以及何种类型的噪声，例如，通过优化目标组合或选择最能服务于下游用途的损坏机制 [Tay et al., 2023]。在扩散建模中，这促使学习时间非齐次的损坏策略，以塑造轨迹，从而获得更好的样本质量或更少的生成步骤。

5.4.3 单纯形与 Logit 动态作为连续松弛方法

我们已经看到，连续时间马尔可夫链（CTMC）的表述将离散扩散过程建模为分布动态。尽管样本路径 $\mathbf{x}(t) \in V^n$ 在分类状态间跳跃，但通过式 (114) 中的 Kolmogorov 方程，边缘分布 $p_t(\cdot)$ 会随时间 t 平滑演化。这一观察启发了我们对语言建模的有用松弛方法：不再将中间状态视为硬标记序列，而是用单纯形上的概率列向量表示每个位置：

$$\mathbf{p}_i(t) \in \Delta^{|V|-1}, \tag{141}$$

$$\mathbf{p}_i(t)[v] = \Pr(x_i(t) = v \mid \mathbf{c}). \tag{142}$$

我们将这些向量聚合为矩阵 $\mathbf{P}(t) = [\mathbf{p}_1(t), \ldots, \mathbf{p}_n(t)] \in \mathbb{R}^{|V| \times n}$，其中每列位于单纯形上。这将离散轨迹转换为单纯形积空间上的连续曲线，从而可以应用 ODE/SDE 工具的同时保持标记的分类特性。

对于生成器为 $\mathbf{Q}^{\text{tok}}(t) \in \mathbb{R}^{|V| \times |V|}$ 的标记级 CTMC，单个位置的边缘分布满足 Kolmogorov 正向方程：

$$\frac{d\mathbf{p}_i(t)}{dt} = \left[\mathbf{Q}^{\text{tok}}(t)\right]^\top \mathbf{p}_i(t), \tag{143}$$

这是单纯形上的一个 ODE。注意 $\mathbf{Q}^{\text{tok}}(t)$ 的行和为零，因此总概率守恒。在噪声过程中各位置独立的假设下，每个 $\mathbf{p}_i(t)$ 都遵循相同的 ODE 演化。这为语言建模提供了 ODE 视角：离散扩散可视为选择由 $\mathbf{Q}^{\text{tok}}(t)$ 诱导的单纯形上线性正向向量场，而非 $\mathbb{R}^d$ 中的高斯 SDE。

直接在概率空间操作可能在单纯形边界附近（例如 one-hot 顶点）存在数值问题。常用替代方案是通过 logit 向量 $\mathbf{z}_i(t) \in \mathbb{R}^{|V|}$ 参数化 $\mathbf{p}_i(t)$：

$$\mathbf{p}_i(t) \quad = \quad \text{Softmax}(\mathbf{z}_i(t)). \tag{144}$$

由此可在 logit 空间定义连续时间动态：

$$\frac{d\mathbf{z}_i(t)}{dt} \quad = \quad \mathbf{v}_\theta\big(\mathbf{z}_i(t), t, \mathbf{c}\big), \tag{145}$$

其中 $\mathbf{v}_\theta$ 是由 Transformer 定义的向量场（各位置共享）。直观上，$\mathbf{z}_i(t)$ 如同连续细化的”软标记”，终端解码 $\mathbf{z}_i(0) \mapsto x_i$ 可通过 $\text{Softmax}(\mathbf{z}_i(0))$ 的 $\arg\max$ 实现。当需要可微采样时，也可使用 Gumbel-Softmax 分布等分类采样的松弛方法 [Jang et al., 2017; Maddison et al., 2017]。

这种 logit 视角阐明了为何连续松弛常与嵌入空间扩散相似，却更接近离散建模：状态保持词汇对齐（每位置维度 $|V|$）而非任意嵌入空间 $\mathbb{R}^d$。它还直接关联到式 (143) 的 CTMC 边缘 ODE：选择的 $\mathbf{Q}^{\text{tok}}(t)$ 决定了 $\mathbf{p}_i(t)$ 的特定演化，可通过 Softmax 映射重新表达为 $\mathbf{z}_i(t)$ 的演化。

如第4节所述，在连续扩散中，逆向时间动态既可表示为涉及分数 $\nabla \log p_t(\mathbf{s})$ 的 SDE，也可表示为概率流 ODE。在离散扩散中，精确的逆向 CTMC 速率取决于概率比 $p_t(u)/p_t(v)$（式 (115)），其作用类似于离散分数。当我们将状态提升至 $\mathbf{p}(t)$ 或 $\mathbf{z}(t)$ 时，可类似地将逆向过程解释为学习从简单先验（如全掩码或均匀分布）流向集中于表示真实文本的 one-hot 顶点附近的分布的向量场。最近的离散分数建模研究表明，可以定义避免显式计算 $p_t(\cdot)$ 但仍产生理论合理的逆向动态的可处理训练目标 [Hoogeboom et al., 2021; Lou et al., 2024]。

总结而言，我们概述单纯形与 logit 松弛的实践优势如下：

- **更高效的 ODE 求解器与更少的步数**。一旦将逆向动力学表达为如式 (145) 所示的 ODE，我们就可以使用自适应 ODE 求解器或粗粒度离散化来减少采样步数，这与连续扩散和流模型中使用的加速策略相呼应（第4.4节）。
- **与流匹配的兼容性**。在单纯形上，流匹配自然地对应于学习将软标记分布向尖锐的、类似数据的分类分布传输的 $\mathbf{v}_\theta$，而无需在训练时依赖显式的基于似然的 ELBO。
- **离散与连续之间的谱系**。掩码扩散（硬 `[MASK]` 状态）与嵌入扩散（连续嵌入）可被视为两个端点。单纯形/逻辑动态占据了一个中间地带。状态是连续的，但仍可直接解释为标记概率，并且到文本的最终投影是定义良好的。

5.4.4 与迭代优化的关联

离散扩散的逆向过程与一系列关于非自回归生成中**迭代优化**的研究密切相关 [Lee et al., 2018]。在迭代优化中，模型并非一次性获得最终序列，而是通过多轮优化迭代，反复生成完整序列并逐步修正其部分内容。例如条件掩码语言模型中的掩码预测方法，该方法从全 `[MASK]` 模板出发，交替执行并行词元预测与不确定位置子集的重新掩码操作，直至收敛或达到固定迭代次数 [Ghazvininejad et al., 2019]。

该方法可表述为与式 (137) 相似的形式。设 $\mathbf{x}^{(r)}$ 表示第 r 轮优化后的序列，$\pi_{\theta,i}^{(r)}(v|\mathbf{x}^{(r)},\mathbf{c}) = p_\theta(x_i^{(r+1)} = v|\mathbf{x}^{(r)},\mathbf{c})$ 为第 r 轮位置 i 的预测分布，则优化步骤可表示为：

$$x_i^{(r+1)} \quad = \quad \begin{cases} x_i^{(r)}, & i \notin \mathcal{U}^{(r)}, \\ \mathrm{Cat}\left(\pi_{\theta,i}^{(r)}(\cdot|\mathbf{x}^{(r)},\mathbf{c})\right), & i \in \mathcal{U}^{(r)}, \end{cases} \tag{146}$$

其中 $\mathcal{U}^{(r)}$ 为第 r 轮待更新位置集合。典型情况下 $\mathcal{U}^{(r)}$ 由低置信度定义，而简单扩散实现中常采用时间调度策略。换言之，掩码扩散采样可视为一种结构化优化过程——其掩码模式随时间 t（即迭代优化中的迭代次数）单调演化，而经典优化方法允许基于模型不确定性的非单调决策（如对已填充词元重新掩码）。这一差异至关重要：单调解掩过程稳定但可能固化早期错误，而非单调优化虽可修正错误却可能因选择规则不明确导致振荡。

近期扩散语言模型已开始显式融合迭代优化中标准的非单调修正行为。一种方法是在推理阶段重新掩码低置信度词元，将扩散采样与类掩码预测的复查机制结合 [Koh et al., 2025; Zhang et al., 2025a]；另一种方法是通过修改前向破坏过程，使逆向模型不仅学习填充空白还需修正可见错误词元。这可通过混合吸收掩码噪声与替换噪声（如均匀替换）实现，从而产生天然支持修正与编辑的逆向过程 [Rütte et al., 2025]。这些方法与第5.4.1节所述方案存在交集，也呼应了我们关于组合不同扩散策略的讨论。

从扩散模型视角亦可解读迭代优化。式 (143)–(145) 中的单纯形与逻辑松弛为迭代优化提供了微分方程视角。若将每次优化步骤视为软词元状态的微小更新，则迭代优化即成为朝向尖锐近单点分布的连续时间传输的显式数值离散化。这与近期离散流匹配 formulations 相连接——后者旨在学习可用较少优化步骤采样同时保持质量的连续时间流 [Gat et al., 2024; Monsefi et al., 2025]。由此观之，扩散建模、基于流的建模与经典迭代优化可视为同一主题的变体，其主要差异在于：中间状态表示方式（硬词元 vs. 软分布）、更新集 $\mathcal{U}$ 选择策略（调度 vs. 置信度）、以及动态过程框架（马尔可夫过程 vs. 通用学习优化算子）。

5.4.5 扩散变换器

离散扩散在自然语言处理领域变得具有竞争力的一个实际原因是其逆向动力学可以通过变换器进行参数化。该架构是当前最具可扩展性的骨干网络，在语言建模领域占据主导地位。术语 扩散变换器（DiT）在视觉领域应用最为广泛，其中用变换器骨干网络替换 U-Net 能够产生强大的缩放行为 [Peebles and Xie, 2023]。详细介绍变换器超出了本文的范围。我们建议感兴趣的读者参阅相关论文 [Vaswani et al., 2017; Xiao and Zhu, 2023]。

在自然语言处理中，变换器提供了一种强大的机制，可以将一个词元序列编码为等长的实值表示序列。根据这些表示的定义和解释方式，此类模型可用于学习从输入词元序列到任意目标的映射。在扩散建模中，变换器必须被条件化于扩散时间 t（或离散索引）。这通常通过将学习到的时间嵌入注入到每一层来实现，例如，通过加性偏置，或与一个时间词元进行交叉注意力。网络的输出可以有多种解释方式，这取决于所选择的参数化方法。在连续空间扩散中，常见的选择包括预测得分 $\nabla_{\mathbf{s}} \log p_t(\mathbf{s})$、加性噪声 $\boldsymbol{\epsilon}$、去噪数据 $\mathbf{s}(0)$，或是线性组合上述变量的速度变量 [Ho et al., 2020a; Salimans and Ho, 2022]。这些参数化方法会导致不同的离散化方案和稳定性特性，但本质上都可以视为学习一个时间索引的向量场，该向量场将概率质量从一个简单的基础分布输送到数据分布。

5.4.6 基于流的视角

鉴于 Transformer 提供的建模灵活性，我们无需局限于使用显式的基于似然的 ELBO（见公式 (122)）来训练扩散模型。一个自然的替代方案是流匹配。在连续设定下，流匹配学习一个时间依赖的向量场，其常微分方程将一个简单的基分布输送到数据分布。对于语言而言，关键困难在于状态定义在离散的乘积空间 V^n 上，因此流必须定义为要么是词元上概率质量的动态（离散状态视角），要么是定义在连续松弛（如概率单纯形或 logit 空间）上的常微分方程（连续状态视角）。最近的**离散流匹配**方法提供了这两种视角，并使它们在现代语言建模的规模下变得实用 [Gat et al., 2024; Campbell et al., 2024; Shaul et al., 2025]。

一个有用的起点是公式 (143) 中已经引入的单纯形表示。我们不将 $\mathbf{x}(t)$ 视为离散的词元序列，而是将中间状态视为每个位置的分类概率

$$\mathbf{P}(t) \quad = \quad [\mathbf{p}_1(t), \ldots, \mathbf{p}_n(t)]^\top \in (\Delta^{|V|-1})^n, \tag{147}$$

其中

$$\sum_{v \in V} \mathbf{p}_i(t)[v] \quad = \quad 1. \tag{148}$$

在这种表示下，流是定义在单纯形乘积空间上的一个常微分方程，

$$\frac{d\mathbf{p}_i(t)}{dt} \quad = \quad \mathbf{v}_\theta(\mathbf{p}_i(t), t, \mathbf{c}), \tag{149}$$

$$\sum_{v \in V} \mathbf{v}_\theta(\mathbf{p}_i(t), t, \mathbf{c})[v] \quad = \quad 0, \tag{150}$$

其中切向约束确保了概率守恒。采样因此成为一个常微分方程积分问题，这与视觉领域类似：给定来自基分布的 $\mathbf{p}(T)$，反向积分公式 (150) 以获得 $\mathbf{p}(0)$，并通过 $\arg\max$ 或分类采样解码出词元。

剩下的任务是指定 $\mathbf{v}_\theta$ 的训练目标。离散流匹配通过定义一族在源分布和数据分布之间插值的概率路径来实现这一点，然后使模型向量场与这些路径诱导的瞬时概率速度相匹配。更具体地说，对于每个训练样本 $\mathbf{x}(0)$，定义 V^n 上或每个词元边缘分布上的条件路径 $q_t(\cdot|\mathbf{x}(0), \mathbf{c})$，并抽取一个带噪样本 $\mathbf{x}(t) \sim q_t(\cdot|\mathbf{x}(0), \mathbf{c})$。流匹配的回归目标是沿着该路径的条件速度，

$$\mathbf{v}^\star(\mathbf{x}(t), t, \mathbf{x}(0), \mathbf{c}) \quad = \quad \frac{d\mathbb{E}\big[\mathbf{1}_{\mathbf{x}(t)}|\mathbf{x}(0), \mathbf{c}\big]}{dt}, \tag{151}$$

其中 $\mathbf{1}_{\mathbf{x}(t)}$ 是 $\mathbf{x}(t)$ 的独热表示。最终的目标函数与连续条件流匹配类似，区别在于状态和速度受到单纯形几何以及端点分类性质的约束。

加噪核可以与之前讨论的扩散语言模型中使用的核非常相似。对于每个位置，我们可以定义一个在干净点质量和基分布 $\bar{p}_i(\cdot)$ 之间的时间依赖混合，

$$q_t(x_i|x_i(0), \mathbf{c}) \quad = \quad \alpha_t \cdot \mathbb{I}(x_i = x_i(0)) + (1 - \alpha_t) \cdot \bar{p}_i(x_i), \tag{152}$$

其中调度 α_t 从 1 递减到 0。这即使在样本路径是离散的情况下，也能在单纯形中诱导出一条平滑的概率路径。

对公式 (152) 关于时间求导，得到目标速度的解析形式

$$\mathbf{v}_i^\star(x_i(t), t, x_i(0), \mathbf{c}) \quad = \quad \dot{\alpha}_t \cdot \left[\mathbf{1}_{x_i(0)} - \bar{p}_i\right], \tag{153}$$

其中 $\dot{\alpha}_t = d\alpha_t/dt$ 表示调度的时间导数。这个简单的线性形式意味着，条件流以 $\dot{\alpha}_t$ 决定的速率，将概率质量从基分布 $\bar{p}_i$ 直接导向目标词元 $x_i(0)$。

为了训练模型，我们需要一个仅依赖于时间 t 时可用信息的条件速度。与标准流匹配一样，训练目标定义为

$$\mathcal{L}_{\text{DFM}}(\theta) \quad = \quad \mathbb{E}_{t,\mathbf{x}(0),\mathbf{x}(t)}\left[\left\|\mathbf{v}_\theta(\mathbf{x}(t), t, \mathbf{c}) - \mathbf{v}^\star(\mathbf{x}(t), t, \mathbf{x}(0), \mathbf{c})\right\|^2\right]. \tag{154}$$

上述单纯形常微分方程视角描述了一条连续的边缘分布曲线，但它没有指定在生成过程中离散样本应如何演化。一个互补的视角是通过一个由时间依赖的生成器矩阵 $\mathbf{Q}^{\text{tok}}(t) \in \mathbb{R}^{|V|\times|V|}$ 参数化的连续时间马尔可夫链来表示离散流。边缘速度 $\mathbf{v}(t)$ 与生成器之间的联系由 Kolmogorov 前向方程给出

$$\frac{d\mathbf{p}_i(t)}{dt} \quad = \quad \left[\mathbf{Q}^{\text{tok}}(t)\right]^\top \mathbf{p}_i(t), \tag{155}$$

其中 $\mathbf{p}_i(t)$ 是表示词元上概率分布的向量。通过为特定目标词元 z 展开这个矩阵乘法，我们可以将速度解释为一个通量平衡方程

$$\mathbf{v}_i(t)[z] \quad = \quad \sum_{y\neq z} \underbrace{\mathbf{p}_i(t)[y] Q^{\text{tok}}_{yz}(t)}_{\text{inflow from } y\to z} - \sum_{y\neq z} \underbrace{\mathbf{p}_i(t)[z] Q^{\text{tok}}_{zy}(t)}_{\text{outflow from } z\to y}. \tag{156}$$

这里，$Q^{\text{tok}}_{yz}(t)$ 表示从词元 y 跳转到 z 的瞬时速率。注意，对角线元素满足 $Q^{\text{tok}}_{yy}(t) = -\sum_{z\neq y} Q^{\text{tok}}_{yz}(t)$ 以确保概率守恒。

虽然公式 (153) 导出的边缘速度 $\mathbf{v}^\star$ 固定了概率的净变化，但它并未唯一确定转移速率 $\mathbf{Q}^{\text{tok}}$。离散流匹配方法通过定义一个以最小噪声实现输运的目标生成器 $\mathbf{Q}^{\star,\text{tok}}$ 来解决这种模糊性。近期工作中一个常见的选择是定义仅允许朝向目标样本 $x_i(0)$ 转移的速率 [Campbell et al., 2024]。具体来说，条件目标速率定义为

$$Q^{\star,\text{tok}}_{yz}(t \mid x_i(0)) \quad = \quad \frac{\dot{\alpha}_t}{1-\alpha_t} \cdot \mathbb{I}(z = x_i(0)), \quad \text{for } y \neq x_i(0). \tag{157}$$

这个生成器驱动任何非目标词元 y 直接跳转到目标 $x_i(0)$，同时保持由 $\mathbf{v}^\star$ 描述的正确边缘演化。

因此，模型被参数化以预测这个生成器矩阵，训练目标也变成了速率匹配损失，而非公式 (154) 中的向量回归。在生成过程中，可以使用 Gillespie 方法 [Gillespie, 1977] 或 Tau-Leaping [Gillespie, 2001] 等算法模拟该过程，并执行离散跳转 $x(t) \to x(t+\Delta t)$，这些跳转在统计上遵循学习到的概率流。

6. 结论与未来方向

在本文中，我们介绍了常微分方程（ODEs）的基本概念，并探讨了其在离散过程（如神经网络架构）和连续过程（如流模型）建模中的应用。我们进一步将讨论扩展到更广泛的微分方程框

架，包括随机微分方程（SDEs），重点阐述了这些数学工具如何定义视觉与语言扩散建模中的生成过程。在此过程中，我们解决了诸如动力系统刚性和标记空间离散扩散复杂性等实际挑战。

尽管微分方程与深度学习的交叉领域已取得显著成果，仍有若干富有前景的方向值得未来探索：

- 一个方向是考虑缩放与数值稳定性问题。随着模型规模扩展至数十亿参数，确保底层常微分方程的数值稳定性变得日益困难。未来的研究可以探索更鲁棒的自适应求解器和正则化技术，以处理大规模架构中的刚性动力学问题，同时避免过高的计算开销。
- 考虑高效的推理方法也很有价值。减少采样步数仍然是实际部署的主要优先事项。轨迹校正、一致性蒸馏和专用高阶求解器等技术为实现仅需一步或少数几步的高保真生成提供了途径。
- 将连续建模应用于文本等离散数据并不直接。探索原生的离散流匹配和更复杂的连续松弛方法，可能催生出更有效的非自回归语言模型，从而在质量上媲美自回归模型。
- 动力系统的视角为模型可解释性提供了独特的透镜。分析固定点、吸引子以及学习轨迹的几何结构，可以更深入地理解神经网络在现实世界问题上的工作原理。
- 将动力系统观点扩展到更广泛的学习任务中是自然而然的，不仅限于本文讨论的问题。例如，将优化过程本身视为一个动力系统，可能有助于将训练稳定性、泛化能力与经典稳定性及扰动理论结果联系起来，并可能提供一个统一框架来理解算法选择（例如，步长调度、动量或隐式更新）如何与模型架构及数据相互作用。

Acknowledgements

This work was supported in part by the National Science Foundation of China (Nos. 62276056 and U24A20334), the Yunnan Fundamental Research Projects (No.202401BC070021), the Yunnan Science and Technology Major Project (No. 202502AD080014), the Fundamental Research Funds for the Central Universities (Nos. N25BSS054 and N25BSS094), and the Program of Introducing Talents of Discipline to Universities, Plan 111 (No.B16009). The authors thank Junxiang Zhang and Hengyu Li for their valuable comments, which helped improve the manuscript.

Appendix A. 推导重球微分方程

回顾带动量的随机梯度下降法，其更新规则定义为

$$v_{k+1} = \mu v_k - \eta \cdot \frac{dL(\theta_k)}{d\theta_k} \tag{158}$$

$$\theta_{k+1} = \theta_k + v_{k+1}, \tag{159}$$

其中 $\mu \in [0, 1)$ 为动量系数，$\eta > 0$ 为学习率。

为推导连续时间极限，我们首先消去动量（或速度）变量 v_k。根据式 (159) 可得 $v_{k+1} = \theta_{k+1} - \theta_k$，这意味着 $v_k = \theta_k - \theta_{k-1}$。将这些表达式代入式 (158) 后，得到关于参数的单一递推关系

$$\theta_{k+1} - \theta_k = \mu(\theta_k - \theta_{k-1}) - \eta \cdot \frac{dL(\theta_k)}{d\theta_k}. \tag{160}$$

重新排列各项以识别有限差分近似，我们得到

$$\underbrace{(\theta_{k+1} - 2\theta_k + \theta_{k-1})}_{\text{加速度项}} + \underbrace{(1-\mu)(\theta_k - \theta_{k-1})}_{\text{速度项}} = -\eta \cdot \frac{dL(\theta_k)}{d\theta_k}. \tag{161}$$

为获得有意义的极限，我们引入时间步长 Δt 使得 $t = k \cdot \Delta t$。根据标准尺度分析 [Polyak, 1964; Su et al., 2016]，假设学习率随步长呈二次方缩放 ($\eta = (\Delta t)^2$)，且动量系数按 $\mu = 1 - \lambda(\Delta t)^2$ 缩放，其中 $\lambda > 0$ 作为阻尼因子。将这些缩放关系代入方程并除以 $(\Delta t)^2$，可得

$$\frac{\theta_{k+1} - 2\theta_k + \theta_{k-1}}{(\Delta t)^2} + \lambda \cdot \frac{\theta_k - \theta_{k-1}}{\Delta t} = -\frac{dL(\theta_k)}{d\theta_k}. \tag{162}$$

当 $\Delta t \to 0$ 时，有限差分商分别收敛于时间变量的二阶和一阶导数。最终，更新序列收敛至重球微分方程

$$\frac{d^2\theta_t}{dt^2} + \lambda \cdot \frac{d\theta_t}{dt} = -\frac{dL(\theta_t)}{d\theta_t}. \tag{163}$$

Appendix B. 从连续性方程推导连续归一化流

这里我们提供一个推导，表明 CNF 中使用的瞬时变量变化公式是描述概率守恒的连续性方程的必然结果。

令 $p_t(\mathbf{x})$ 表示时间 t 的概率密度函数，令 $f(\mathbf{x}, \theta, t)$ 表示由神经网络生成的连续向量场。注意，为符号简洁起见，这里我们使用 $\mathbf{x}$ 表示时间相关的轨迹 $\mathbf{x}(t)$。概率质量的守恒由连续性方程描述（欧拉视角）

$$\frac{\partial p_t(\mathbf{x})}{\partial t} + \nabla \cdot (p_t(\mathbf{x}) f(\mathbf{x}, \theta, t)) = 0, \tag{164}$$

其中 $\nabla\cdot$ 表示关于 $\mathbf{x}$ 的散度算子。

利用向量微积分恒等式 $\nabla\cdot(\phi\mathbf{A})=(\nabla\phi)\cdot\mathbf{A}+\phi(\nabla\cdot\mathbf{A})$，我们展开散度项如下

$$\frac{\partial p_t(\mathbf{x})}{\partial t}+\nabla p_t(\mathbf{x})\cdot f(\mathbf{x},\theta,t)+p_t(\mathbf{x})\nabla\cdot f(\mathbf{x},\theta,t) \quad = \quad 0. \tag{165}$$

虽然式 (165) 描述了固定空间点的密度演化，但 CNF 通过追踪运动粒子轨迹上的密度来运作（拉格朗日视角）。该轨迹由 ODE $\frac{d\mathbf{x}}{dt}=f(\mathbf{x}(t),\theta,t)$ 定义。

为了连接这两种视角，我们计算沿该轨迹的概率密度的全时间导数（也称为物质导数）。首先，对 $p_t(\mathbf{x})$ 应用多元链式法则，我们得到一般形式

$$\frac{dp_t(\mathbf{x})}{dt} \quad = \quad \frac{\partial p_t(\mathbf{x})}{\partial t}+\nabla p_t(\mathbf{x})\cdot\frac{d\mathbf{x}}{dt}. \tag{166}$$

这里，从流体力学视角看，偏导数 $\frac{\partial p_t}{\partial t}$ 可视为固定空间位置处的局部变化率，而全导数 $\frac{dp_t}{dt}$ 可视为粒子沿其轨迹运动时所经历的总变化。

接下来，将流动的特定 ODE $\frac{d\mathbf{x}}{dt}=f(\mathbf{x},\theta,t)$ 代入式 (166)，我们得到用向量场 f 表示的物质导数

$$\frac{dp_t(\mathbf{x})}{dt} \quad = \quad \frac{\partial p_t(\mathbf{x})}{\partial t}+\nabla p_t(\mathbf{x})\cdot f(\mathbf{x},\theta,t). \tag{167}$$

现在，将式 (167) 中的物质导数表达式代入展开的连续性方程式 (165)，我们可以识别并替换前两项

$$\underbrace{\frac{\partial p_t}{\partial t}+\nabla p_t\cdot f}_{\frac{dp_t}{dt}}+p_t(\nabla\cdot f) \quad = \quad 0, \tag{168}$$

为清晰起见，此处省略了 p_t 和 f 的参数。这简化了关系式得到

$$\frac{dp_t(\mathbf{x})}{dt}=-p_t(\mathbf{x})\nabla\cdot f(\mathbf{x},\theta,t). \tag{169}$$

为了推导标准的 CNF 公式，我们考虑对数密度 $\log p_t(\mathbf{x})$。使用导数规则 $\frac{d}{dt}\log u(t)=\frac{1}{u(t)}\frac{du(t)}{dt}$，我们将两边同时除以 $p_t(\mathbf{x})$，得到

$$\frac{d\log p_t(\mathbf{x})}{dt} \quad = \quad -\nabla\cdot f(\mathbf{x},\theta,t). \tag{170}$$

最后，认识到向量场的散度是其雅可比矩阵的迹，即 $\nabla\cdot f=\mathrm{Tr}\left(\frac{\partial f}{\partial\mathbf{x}}\right)$，我们便得到了瞬时变量变化公式（即式 (73)）

$$\frac{d\log p_t(\mathbf{x})}{dt} \quad = \quad -\mathrm{Tr}\left(\frac{\partial f(\mathbf{x},\theta,t)}{\partial\mathbf{x}}\right). \tag{171}$$

Appendix C. 从 SDE 推导概率流 ODE

在此，我们提供第 4.3 节中介绍的概率流 ODE 的详细推导。我们证明，对于给定的扩散 SDE，存在一个确定性 ODE，其轨迹诱导出完全相同的边缘概率密度 $\{p_t\}_{t=0}^{T}$。

考虑定义的前向 SDE 为

$$d\mathbf{x}(t) \quad = \quad f(\mathbf{x}(t),t)dt + g(t)d\mathbf{w}(t), \tag{172}$$

其中 $f(\mathbf{x}(t),t)$ 是漂移系数，$g(t)$ 是扩散系数，$\mathbf{w}(t)$ 是标准维纳过程。为简化符号，下文省略 $\mathbf{x}(t)$ 中的参数 t。

状态变量 $\mathbf{x}$ 的概率密度函数 $p_t(\mathbf{x})$ 的时间演化由 Fokker-Planck 方程（也称为 Kolmogorov 前向方程）控制：

$$\frac{\partial p_t(\mathbf{x})}{\partial t} \quad = \quad -\nabla\cdot[f(\mathbf{x},t)p_t(\mathbf{x})] + \frac{1}{2}g(t)^2\Delta p_t(\mathbf{x}), \tag{173}$$

其中 $\nabla\cdot$ 表示散度算子，$\Delta = \nabla\cdot\nabla$ 是拉普拉斯算子。等式右边的第一项表示由于漂移引起的概率质量平流，第二项表示由随机噪声引起的扩散。

为了推导等效的 ODE，我们将方程 (173) 重写为连续性方程的形式，该形式描述确定性系统的密度演化（另见附录 B）。连续性方程仅涉及一阶导数，其形式为

$$\frac{\partial p_t(\mathbf{x})}{\partial t} \quad = \quad -\nabla\cdot[\tilde{f}(\mathbf{x},t)p_t(\mathbf{x})], \tag{174}$$

其中 $\tilde{f}(\mathbf{x},t)$ 对应于确定性 ODE 的向量场。

关键步骤是将扩散项 $\Delta p_t(\mathbf{x})$（涉及二阶导数）表示为一阶项的散度。我们利用恒等式 $\nabla p_t(\mathbf{x}) = p_t(\mathbf{x})\nabla\log p_t(\mathbf{x})$ 将拉普拉斯算子重写为

$$\begin{aligned}\Delta p_t(\mathbf{x}) &= \nabla\cdot(\nabla p_t(\mathbf{x})) \\ &= \nabla\cdot(p_t(\mathbf{x})\nabla\log p_t(\mathbf{x})).\end{aligned} \tag{175}$$

将此恒等式代回 Fokker-Planck 方程 (173)，我们可以将各项组合在单个散度算子下，如下所示

$$\begin{aligned}\frac{\partial p_t(\mathbf{x})}{\partial t} &= -\nabla\cdot[f(\mathbf{x},t)p_t(\mathbf{x})] + \frac{1}{2}g(t)^2\nabla\cdot[p_t(\mathbf{x})\nabla\log p_t(\mathbf{x})] \\ &= -\nabla\cdot\left[f(\mathbf{x},t)p_t(\mathbf{x}) - \frac{1}{2}g(t)^2 p_t(\mathbf{x})\nabla\log p_t(\mathbf{x})\right] \\ &= -\nabla\cdot\left[\left(f(\mathbf{x},t) - \frac{1}{2}g(t)^2\nabla\log p_t(\mathbf{x})\right)p_t(\mathbf{x})\right].\end{aligned} \tag{176}$$

将此结果与方程 (174) 中的连续性方程进行比较，我们发现，如果我们如下定义有效的确定性向量场 $\tilde{f}(\mathbf{x},t)$，则两个方程匹配：

$$\tilde{f}(\mathbf{x},t) \quad = \quad f(\mathbf{x},t) - \frac{1}{2}g(t)^2\nabla\log p_t(\mathbf{x}). \tag{177}$$

因此，该确定性 ODE 产生的边缘分布 $p_t(\mathbf{x})$ 与原始 SDE 定义的随机过程完全相同。该 ODE 由下式给出

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla \log p_t(\mathbf{x}). \tag{178}$$

参考文献

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In The Eleventh International Conference on Learning Representations, 2023.

Brian D.O. Anderson. Reverse-time diffusion equation models. Stochastic Processes and their Applications, 12(3):313–326, 1982.

Tom M Apostol. Calculus, Volume 1. John Wiley & Sons, 1991.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. Advances in neural information processing systems, 34:17981–17993, 2021.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In International Conference on Learning Representations, 2019.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. Machine learning for data science handbook: data mining and knowledge discovery handbook, pages 353–374, 2023.

George Keith Batchelor. An introduction to fluid dynamics. Cambridge university press, 2000.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In International conference on machine learning, pages 573–582. PMLR, 2019.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

Richard L. Burden, J. Douglas Faires, and Annette M. Burden. Numerical Analysis. Cengage Learning, 2015.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In International Conference on Machine Learning, pages 5453–5512. PMLR, 2024.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. Advances in neural information processing systems, 31, 2018.

Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. Advances in neural information processing systems, 32, 2019.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In Proceedings of International Conference on Learning Representations, 2019.

EA Coddington. Theory of ordinary differential equations. McGraw-Hill Book Company, 1955.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. IEEE transactions on pattern analysis and machine intelligence, 45(9): 10850–10869, 2023.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In International Conference on Learning Representations, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, 2019.

Sander Dieleman. Perspectives on diffusion, 2023. URL https://sander.ai/2023/07/20/perspectives.html.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In International Conference on Learning Representations, 2017.

William Dunham. The calculus gallery: Masterpieces from Newton to Lebesgue. Princeton University Press, 2005.

CH Jr Edwards. The historical development of the calculus. Springer Science & Business Media, 1994.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In Forty-first international conference on machine learning, 2024.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. Advances in Neural Information Processing Systems, 37:133345–133385, 2024.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6112–6121, 2019.

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. The journal of physical chemistry, 81(25):2340–2361, 1977.

Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. The Journal of chemical physics, 115(4):1716–1733, 2001.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In The Eleventh International Conference on Learning Representations, 2023.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. Machine Learning, 110(2):393–416, 2021.

Or Greenberg. Demystifying flux architecture. arXiv preprint arXiv:2507.09595, 2025.

J Gu, J Bradbury, C Xiong, VOK Li, and R Socher. Non-autoregressive neural machine translation. In International Conference on Learning Representations (ICLR), 2018.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. Advances in neural information processing systems, 32, 2019.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun

Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, Tao Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nat., 645(8081):633–638, 2025. doi: 10.1038/S41586-025-09422-Z. URL https://doi.org/10.1038/s41586-025-09422-z.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. Inverse problems, 34(1), 2017.

Ernst Hairer and Gerhard Wanner. Solving Ordinary Differential Equations II. Cambridge university press, 1996.

Philip Hartman. Ordinary differential equations. SIAM, 2002.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Morris W Hirsch, Stephen Smale, and Robert L Devaney. Differential equations, dynamical systems, and an introduction to chaos. Academic press, 2013.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020a.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020b.

Emiel Hoogeboom, Jorn Peters, Rianne Van Den Berg, and Max Welling. Integer discrete flows and lossless compression. Advances in Neural Information Processing Systems, 32, 2019.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. Advances in neural information processing systems, 34:12454–12465, 2021.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(4), 2005.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In International Conference on Learning Representations, 2017.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. Transactions of the association for computational linguistics, 8:64–77, 2020.

Ioannis Karatzas and Steven Shreve. Brownian motion and stochastic calculus. springer, 2014.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. Advances in neural information processing systems, 35: 26565–26577, 2022.

Diederik P Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. Foundations and Trends® in Machine Learning, 12(4):307–392, 2019.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. Advances in neural information processing systems, 31, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. Advances in neural information processing systems, 29, 2016.

Peter E Kloeden and RA Pearson. The numerical solution of stochastic differential equations. The ANZIAM Journal, 20(1):8–12, 1977.

Peter E. Kloeden and Eckhard Platen. Numerical Solution of Stochastic Differential Equations. Springer Berlin, 1992.

Hyukhun Koh, Minha Jhang, Dohyung Kim, Sangmook Lee, and Kyomin Jung. Conditional [mask] discrete diffusion language model. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 8910–8934, 2025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In International Conference on Learning Representations, 2020.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. Predicting structured data, 1, 2006.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182, 2018.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, 2020.

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, Jingbo Zhu, Xuebo Liu, and Min Zhang. Ode transformer: An ordinary differential equation-inspired model for sequence generation. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8335–8351, 2022a.

Bei Li, Tong Zheng, Rui Wang, Jiahao Liu, Junliang Guo, Xu Tan, Tong Xiao, JingBo Zhu, Jingang Wang, and Xunliang Cai. Predictor-corrector enhanced transformers with exponential moving average coefficient learning. Advances in Neural Information Processing Systems, 37: 20358–20382, 2024.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models, 2025.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. Advances in neural information processing systems, 35:4328–4343, 2022b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In The Eleventh International Conference on Learning Representations, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. arXiv preprint arXiv:2412.06264, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding,

Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In International Conference on Learning Representations, 2022.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In The Eleventh International Conference on Learning Representations, 2023.

Xinyu Liu, Bei Li, Jiahao Liu, Junhao Ruan, Kechen Jiao, Hongyin Tang, Jingang Wang, Tong Xiao, and Jingbo Zhu. Iiet: Efficient numerical transformer via implicit iterative euler method. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 8955–8969, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In Proceedings of the 41st International Conference on Machine Learning, pages 32819–32848, 2024.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In The Thirteenth International Conference on Learning Representations, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. Advances in neural information processing systems, 35:5775–5787, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. Machine Intelligence Research, pages 1–22, 2025.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. In ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.

Calvin Luo. Understanding diffusion models: A unified perspective. arXiv, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. arXiv preprint arXiv:2310.04378, 2023a.

Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick Von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. arXiv preprint arXiv:2311.05556, 2023b.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In International Conference on Learning Representations, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Proceedings of the International Conference on Learning Representations (ICLR 2013), 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, pages 3111–3119, 2013b.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In International Conference on Learning Representations, 2018.

Amin Karimi Monsefi, Nikhil Bhendawade, Manuel Rafael Ciosici, Dominic Culver, Yizhe Zhang, and Irina Belousova. Fs-dfm: Fast and accurate long text generation with few-step diffusion language models. arXiv preprint arXiv:2509.20624, 2025.

Behnam Neyshabur. Implicit regularization in deep learning. arXiv preprint arXiv:1709.01953, 2017.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. arXiv preprint arXiv:2502.09992, 2025.

James R Norris. Markov chains. Cambridge university press, 1998.

Bernt Oksendal. Stochastic differential equations: an introduction with applications. Springer Science & Business Media, 2013.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. Advances in neural information processing systems, 30, 2017.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In Proceedings of the IEEE/CVF international conference on computer vision, pages 4195–4205, 2023.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. Ussr computational mathematics and mathematical physics, 4(5):1–17, 1964.

Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks, 12(1):145–151, 1999.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020.

Machel Reid, Vincent J Hellendoorn, and Graham Neubig. Diffuser: Discrete diffusion via edit-based reconstruction. arXiv preprint arXiv:2210.16886, 2022.

Christian P Robert, George Casella, and George Casella. Monte Carlo statistical methods, volume 2. Springer, 1999.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. In Forty-second International Conference on Machine Learning, 2025.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. Advances in Neural Information Processing Systems, 37:130136–130184, 2024.

Shreshth Saini, Shashank Gupta, and Alan C Bovik. Rectified-cfg++ for flow based models. arXiv preprint arXiv:2510.07631, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In International Conference on Learning Representations, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In European Conference on Computer Vision, pages 87–103. Springer, 2024.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow matching with general discrete paths: A kinetic-optimal perspective. In The Thirteenth International Conference on Learning Representations, 2025.

Hui Shen, Jingxuan Zhang, Boning Xiong, Rui Hu, Shoufa Chen, Zhongwei Wan, Xin Wang, Yu Zhang, Zixuan Gong, Guangyin Bao, Chaofan Tao, Yongfeng Huang, Ye Yuan, and Mi Zhang. Efficient diffusion models: A survey. arXiv preprint arXiv:2502.06805, 2025.

Yang Song. Generative modeling by estimating gradients of the data distribution, 2021. URL https://yang-song.net/blog/2021/score/.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems, 32, 2019.

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, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In International Conference on Machine Learning, pages 32211–32252. PMLR, 2023.

Michael Spivak. Calculus. Cambridge University Press, 2006.

Josef Stoer, Roland Bulirsch, R Bartels, Walter Gautschi, and Christoph Witzgall. Introduction to numerical analysis. Springer, 1980.

Gilbert Strang. On the construction and comparison of difference schemes. SIAM journal on numerical analysis, 5(3):506–517, 1968.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation. arXiv preprint arXiv:2211.04236, 2022.

Weijie Su, Stephen Boyd, and Emmanuel J Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. Journal of Machine Learning Research, 17 (153):1–43, 2016.

Terence Tao. An introduction to measure theory, volume 126. American Mathematical Society, 2011.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying language learning paradigms. In The Eleventh International Conference on Learning Representations, 2023.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. Advances in Neural Information Processing Systems, 32, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Pascal Vincent. A connection between score matching and denoising autoencoders. Neural computation, 23(7):1661–1674, 2011.

Fu-Yun Wang, Zhaoyang Huang, Alexander Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, et al. Phased consistency models. Advances in neural information processing systems, 37:83951–84009, 2024.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. Multi-layer representation fusion for neural machine translation. In Proceedings of the 27th international conference on computational linguistics, pages 3015–3026, 2018.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1810–1822, 2019.

Tong Xiao and Jingbo Zhu. Introduction to transformers: an nlp perspective. arXiv preprint arXiv:2311.17633, 2023.

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(10):11407–11427, 2023.

Hanshu Yan, Xingchao Liu, Jiachun Pan, Jun Hao Liew, qiang liu, and Jiashi Feng. PeRFlow: Piecewise rectified flow as universal plug-and-play accelerator. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. ACM computing surveys, 56(4):1–39, 2023.

Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=HHbRxoDTxE.

Xiaofeng Yang, Chen Cheng, Xulei Yang, Fayao Liu, and Guosheng Lin. Text-to-image rectified flow as plug-and-play priors. In The Thirteenth International Conference on Learning Representations, 2025a.

Yicun Yang, Cong Wang, Shaobo Wang, Zichen Wen, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Diffusion llm with native variable generation lengths: Let [eos] lead the way, 2025b.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In NeurIPS 2022 Workshop on Score-Based Methods, 2022.

Shuibai Zhang, Fred Zhangzhi Peng, Yiheng Zhang, Jin Pan, and Grigorios G Chrysos. Corrective diffusion language models. arXiv preprint arXiv:2512.15596, 2025a.

Xinxi Zhang, Shiwei Tan, Quang Nguyen, Quan Dao, Ligong Han, Xiaoxiao He, Tunyu Zhang, Alen Mrdovic, and Dimitris Metaxas. Flow straighter and faster: Efficient one-step generative modeling via meanflow on rectified trajectories. arXiv preprint arXiv:2511.23342, 2025b.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. Advances in Neural Information Processing Systems, 36:49842–49869, 2023.

Jianbin Zheng, Minghui Hu, Zhongyi Fan, Chaoyue Wang, Changxing Ding, Dacheng Tao, and Tat-Jen Cham. Trajectory consistency distillation. CoRR, 2024.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. DPM-solver-v3: Improved diffusion ODE solver with empirical model statistics. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=9fWKExmKa0.

Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7777–7786, 2024.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. In International Conference on Learning Representations, 2020.