底本：arXiv:2407.15595v2，2024-11-05，英文 38 页。完整正文、附录 A–I、图表、代码和生成样例均保留。翻译、排版与译注为本次改编；版权归原作者。许可：CC BY 4.0。

阅读路径：先读第 2 节与表 1，再读安全采样（B）、理论证明（E）及评估协议（G）。公式已本地展开；表格和代码由 PDF 精确裁取，以修复官方 HTML 的斜杠数值和百分号丢失。定义、定理、命题与证明统一采用英文印本编号（定义 1；定理 2、3、4、7；命题 5、6、8），附录小节仍为 E.1–E.10；原 HTML 锚点保留，源页码见每段。

Discrete Flow Matching

离散流匹配

Itai Gat Affiliation: Meta AI, FAIR Tal Remez Affiliation: Meta AI, FAIR Neta Shaul Affiliation: Weizmann Institute Felix Kreuk Affiliation: Meta AI, FAIR Ricky T. Q. Chen Affiliation: Meta AI, FAIR Gabriel Synnaeve Affiliation: Meta AI, FAIR Yossi Adi Affiliation: Meta AI, FAIR
Yaron Lipman Affiliation: Meta AI, FAIR

Abstract

摘要

Despite Flow Matching and diffusion models having emerged as powerful generative paradigms for continuous variables such as images and videos, their application to high-dimensional discrete data, such as language, is still limited. In this work, we present Discrete Flow Matching, a novel discrete flow paradigm designed specifically for generating discrete data. Discrete Flow Matching offers several key contributions: (i) it works with a general family of probability paths interpolating between source and target distributions; (ii) it allows for a generic formula for sampling from these probability paths using learned posteriors such as the probability denoiser (x-prediction) and noise-prediction (ϵ-prediction); (iii) practically, focusing on specific probability paths defined with different schedulers improves generative perplexity compared to previous discrete diffusion and flow models; and (iv) by scaling Discrete Flow Matching models up to 1.7B parameters, we reach 6.7% Pass@1 and 13.4% Pass@10 on HumanEval and 6.7% Pass@1 and 20.6% Pass@10 on *1-shot* MBPP coding benchmarks. Our approach is capable of generating high-quality discrete data in a non-autoregressive fashion, significantly closing the gap between autoregressive models and discrete flow models.

虽然流匹配与扩散模型已成为图像、视频等连续变量的强大生成范式，但在语言等高维离散数据上的应用仍然有限。本文提出离散流匹配（Discrete Flow Matching），一种专门生成离散数据的新离散流范式。主要贡献包括：（i）适用于连接源分布与目标分布的一般概率路径族；（ii）借助学得的概率去噪器（x 预测）、噪声预测器（ε 预测）等后验，提供沿这些路径采样的通用公式；（iii）在实践中，选用不同调度器定义的特定概率路径，相比已有离散扩散和流模型改善了生成困惑度；（iv）将模型扩展到 17 亿参数，在 HumanEval 上取得 6.7% Pass@1、13.4% Pass@10，在单样本提示的 MBPP 编程基准上取得 6.7% Pass@1、20.6% Pass@10。本方法以非自回归方式生成高质量离散数据，显著缩小了自回归模型与离散流模型之间的差距。

1 Introduction
1 引言

S002 · Original / 原文 · PDF p.1 · S1.p1.1

Despite the remarkable success of diffusion and flow models in generating continuous spatial signals such as images *(Ho et al., 2020; Rombach et al., 2022; Esser et al., 2024)* and videos *(Singer et al., 2022; Blattmann et al., 2023)*, their performance still falters when applied to discrete sequential data compared to autoregressive models. Recent progress in adapting diffusion and flow models to the discrete setting has been made via mostly two approaches: embedding the discrete data in continuous space and applying continuous diffusion *(Dieleman et al., 2022; Stark et al., 2024)* or designing diffusion or flow processes over discrete state spaces *(Austin et al., 2021a; Campbell et al., 2022)*.

尽管扩散和流模型在生成图像（Ho 等，2020；Rombach 等，2022；Esser 等，2024）、视频（Singer 等，2022；Blattmann 等，2023）这类连续空间信号上取得了显著成功，但应用于离散序列数据时，其性能相对自回归模型仍显不足。近期将扩散和流推广到离散情形的进展主要来自两条路线：把离散数据嵌入连续空间再进行连续扩散（Dieleman 等，2022；Stark 等，2024），或直接在离散状态空间上设计扩散或流过程（Austin 等，2021a；Campbell 等，2022）。

S003 · Original / 原文 · PDF p.1 · S1.p2.1

In this paper, we pursue the discrete flow approach of *Campbell et al. (2024)* and introduce Discrete Flow Matching, a theoretical framework and algorithmic methodology for discrete flow models that yields a state-of-the-art discrete non-autoregressive generative approach. Surprisingly, Discrete Flow Matching exhibits similarities with the continuous Flow Matching *(Lipman et al., 2022)* approach proposed for continuous signals. Notably, its *generating probability velocity*, employed in the sampling algorithm, is identical in form to its continuous counterpart. Additionally, Discrete Flow Matching offers the following advancements and simplifications over prior methods: It encompasses a more comprehensive family of probability paths transforming source (noise) distributions into target (data) distributions, accommodating arbitrary source-target couplings and time-dependent schedulers. Furthermore, it provides a unified formulation for the generating probability velocity directly expressed in terms of the learned posteriors and schedulers, along with a unified and general theory and algorithm for corrector sampling and iterations. In practice, we observe that path and corrector schedulers are pivotal, and their proper tuning leads to substantial improvements in generation quality. We have trained a 1.7B parameter Discrete Flow Matching model on the same data mix as in Llama-2 *(Touvron et al., 2023)* and CodeLlama *(Roziere et al., 2023)*, achieving 6.7% Pass@1 and 13.4% Pass@10 on HumanEval and 6.7% Pass@1 and 20.6% Pass@10 on *1-shot* MBPP coding benchmarks; Figure 1 shows some code generation examples. In conditional text generation our model produces texts with a generative perplexity score of 9.7 as measured by the Llama-3 8B model, surpassing a 1.7B autoregressive model that achieves 22.3 and not far from the Llama-2 7B model that achieves 8.3 in generative perplexity score. We strongly believe that Discrete Flow Matching represents a significant step in bridging the performance gap between discrete diffusion and autoregressive models, and that further enhancements are possible by exploring the vast design space that Discrete Flow Matching has to offer.

本文沿用 Campbell 等（2024）的离散流路线，提出离散流匹配：它为离散流模型提供理论框架与算法方法，并得到当时先进的离散非自回归生成方法。令人意外的是，离散流匹配与为连续信号提出的连续流匹配（Lipman 等，2022）存在相似性，尤其是采样算法使用的生成概率速度与连续情形具有完全相同的形式。相较已有方法，本框架进一步扩展并简化了设计：用更全面的概率路径族把源（噪声）分布变为目标（数据）分布，容许任意源—目标耦合和随时间变化的调度器；给出以学得后验和调度器直接表示的统一生成概率速度公式，并统一校正采样与校正迭代的理论、算法。实践表明，路径调度器与校正调度器非常关键，恰当调节可明显改善生成质量。我们使用与 Llama-2（Touvron 等，2023）、CodeLlama（Roziere 等，2023）相同的混合数据训练了 17 亿参数模型，取得 HumanEval 的 6.7% Pass@1、13.4% Pass@10，以及单样本提示 MBPP 的 6.7% Pass@1、20.6% Pass@10；图 1 展示若干代码生成例子。条件文本生成中，以 Llama-3 8B 评估的生成困惑度为 9.7，优于一个 17 亿参数自回归模型的 22.3，并接近 Llama-2 7B 的 8.3。我们认为离散流匹配是缩小离散扩散与自回归模型性能差距的重要一步，进一步探索其广阔设计空间还可带来改进。

```python
def fib(n: int):
    """Return n-th Fibonacci
    number.
    >>> fib(10)
    55
    >>> fib(1)
    1
    >>> fib(8)
    21
    """
    if n < 1: return 0
    if n < 2: return 1

    return fib(n-1) + fib(n-2)
```

```python
def find_position_of_value(arr, x):
    low, mid = 0, 0
    high = len(arr) - 1

    while high >= low:
        mid = (high + low) // 2

        # If x is greater
        if arr[mid] < x:
            low = mid + 1
        # If x is smaller
        elif arr[mid] > x:
            high = mid - 1
        else:
            return mid
    return -1
```

```python
def binary_search(arr, x):
    start = 0
    end = len(arr)-1

    # While performing binary search
    while start <= end:
        mid = (start + end) // 2
        # If x is greater
        if arr[mid] < x:
            start = mid + 1
        # If x is smaller
        elif arr[mid] > x:
            end = mid - 1
        else:
            return mid
    return -1
```

源图 / 表 / 代码：英文 PDF 第 2 页 · S1.F1

Figure 1: Code generation examples using Discrete Flow Matching. Code condition is marked in gray, model generation is marked in yellow. Left sub-figure presents the standard left-to-right prompting; Middle and Right sub-figures, presents complex infilling setup.

图 1：离散流匹配的代码生成示例。灰色为给定代码条件，黄色为模型生成。左图为标准从左到右提示，中、右图为复杂代码填空。

2 Discrete Flow Matching

2 离散流匹配

2.1 Setup and notations

2.1 设置与记号

S004 · Original / 原文 · PDF p.2 · S2.SS1.p1.1

In discrete sequence modeling, we denote a sequence x as an array of N elements $(x^1, x^2, \ldots, x^N)$. Each element, or token, within this sequence is selected from a vocabulary of size d. Consequently, the entire set of possible sequences is given by $\mathcal{D} = [d]^N$, where $[d] = \{1, \ldots, d\}$. A random variable taking values in the space $\mathcal{D}$ is denoted by X and its corresponding probability mass function (PMF) is $P(X = x)$. For simplicity, throughout the paper, we sometimes omit the random variable X and use $p(x)$ to denote the PMF.

在离散序列建模中，序列 x 表示为含 N 个元素的数组 $(x^1, x^2, \ldots, x^N)$。每个元素或词元（token）从大小为 d 的词表中选取。因此所有可能序列构成 $\mathcal{D} = [d]^N$，其中 $[d] = \{1, \ldots, d\}$。取值于 D 的随机变量记为 X，其概率质量函数（PMF）为 P(X=x)。为简便，全文有时省略随机变量 X，直接用 p(x) 表示 PMF。

S005 · Original / 原文 · PDF p.2 · S2.SS1.p2.1

To describe marginalization properties, we denote $p(x^i)$ the x^i marginal of p, i.e., $p(x^i) = \sum_{x^{\bar{i}}} p(x)$, where $x^{\bar{i}} = (\ldots, x^{i-1}, x^{i+1}, \ldots) \in [d]^{N-1}$ are all the arguments excluding i. Similarly, $p(x^{\bar{i}}) = \sum_{x^i} p(x)$, and $x^i \in [d]$. A useful PMF is the delta function, δ_y, $y \in \mathcal{D}$, which is defined by

为描述边缘化，记 $p(x^i)$ 为 p 关于第 i 个分量的边缘分布，即 $p(x^i) = \sum_{x^{\bar{i}}} p(x)$；其中 $x^{\bar{i}} = (\ldots, x^{i-1}, x^{i+1}, \ldots) \in [d]^{N-1}$ 表示除 i 外的全部分量。同样，$p(x^{\bar{i}}) = \sum_{x^i} p(x)$，且 $x^i \in [d]$。一个有用的 PMF 是 delta 函数 δ_y（$y \in \mathcal{D}$），定义如下。

$$\delta_y(x) = \prod_{i=1}^{N} \delta_{y^i}(x^i), \text{ where } \delta_{y^i}(x^i) = \begin{cases} 1 & x^i = y^i \\ 0 & x^i \neq y^i \end{cases}. \tag{1}$$

S006 · Original / 原文 · PDF p.2 · S2.SS1.p2.2

With the marginal notation $\delta_y(x^i) = \delta_{y^i}(x^i)$ and $\delta_y(x^{\bar{i}}) = \delta_{y^{\bar{i}}}(x^{\bar{i}}) = \prod_{j \neq i} \delta_{y^j}(x^j)$ which simplifies notation.

使用边缘记号，$\delta_y(x^i) = \delta_{y^i}(x^i)$，以及 $\delta_y(x^{\bar{i}}) = \delta_{y^{\bar{i}}}(x^{\bar{i}}) = \prod_{j \neq i} \delta_{y^j}(x^j)$，可以简化书写。

2.2 Source and target distributions

2.2 源分布与目标分布

S007 · Original / 原文 · PDF p.2 · S2.SS2.p1.1

In discrete generative models our goal is to transform source samples $X_0 \sim p$ to target samples $X_1 \sim q$. Our training data, consist of pairs X_0 and X_1 that are sampled from a joint distribution $\pi(x, y)$, satisfying the marginals constraints $p(x) = \sum_{y \in \mathcal{D}} \pi(x, y), q(y) = \sum_{x \in \mathcal{D}} \pi(x, y)$, i.e.,

离散生成模型旨在把源样本 $X_0 \sim p$ 转换为目标样本 $X_1 \sim q$。训练数据是从联合分布 $\pi(x, y)$ 采样的 X_0, X_1 样本对，该分布满足边缘约束 $p(x) = \sum_{y \in \mathcal{D}} \pi(x, y)$、$q(y) = \sum_{x \in \mathcal{D}} \pi(x, y)$，即下式。

$$(X_0, X_1) \sim \pi(X_0, X_1). \tag{2}$$

S008 · Original / 原文 · PDF p.2 · S2.SS2.p1.2

In the simplest case, the training pairs X_0 and X_1 are sampled independently from the source and target distributions respectively,

最简单的情形，是分别从源分布和目标分布独立抽取训练样本 X_0, X_1。

$$(X_0, X_1) \sim p(X_0)q(X_1). \tag{3}$$

S009 · Original / 原文 · PDF p.2 · S2.SS2.p1.3

Example: source and couplings. Common instantiations of source distribution p are: (i) adding a special token value often referred to as a 'mask' or 'dummy' token, denoted here by m, and setting the source distribution to be all-mask sequences, i.e., $p(x) = \delta_m(x)$; and (ii) using uniform distribution over $\mathcal{D}$, which is equivalent to drawing each x^i independently to be some value in $[d]$ with equal probability, denoted $p(x) = p_{\text{u}}(x)$. In this paper we focus mainly on (i). We further consider two choices of couplings π: Independent coupling, which we call unconditional coupling (U-coupling), $\pi(x_0, x_1) = p(x_0)q(x_1)$. A random sample that realizes this choice have the form

例：源分布与耦合。常见源分布 p 有：（i）增加一个称为 mask 或 dummy 的特殊词元，此处记为 M（M 指原文 m，即双线体小写 m 所表示的同一掩码词元），并把源分布设为全掩码序列，即 $p(x) = \delta_M(x)$；（ii）在 D 上取均匀分布，相当于每个 x^i 独立地等概率选取 [d] 中的一个值，记为 $p(x) = p_{\text{u}}(x)$。本文主要研究（i）。进一步考察两类耦合 π：独立耦合称为无条件耦合（U-coupling），$\pi(x_0, x_1) = p(x_0)q(x_1)$。实现这一耦合的随机样本具有下式形式。

$$(X_0, X_1) = \Big((m, \ldots, m), X_1\Big), \tag{4}$$

S010 · Original / 原文 · PDF p.3 · S2.SS2.p1.4

where $X_1 \sim q(X_1)$ is a random sample from the training set. The second choice of coupling $\pi(x_0, x_1) = p(x_0|x_1)q(x_1)$, which we find improves conditional sampling, partially masks inputs with samples of the form

其中 $X_1 \sim q(X_1)$ 是训练集中的随机样本。第二种耦合 $\pi(x_0, x_1) = p(x_0|x_1)q(x_1)$ 对输入进行部分掩码，样本如下；我们发现它能改善条件采样。

$$(X_0, X_1) = (\mathbb{I} \odot X_1 + (\mathbf{1} - \mathbb{I}) \odot (m, \ldots, m), X_1), \tag{5}$$

S011 · Original / 原文 · PDF p.3 · S2.SS2.p1.5

where $X_1 \sim q(X_1)$ and $\mathbb{I} \in \{0, 1\}^N$ is a random variable indicating the conditioning, $\odot$ denotes the entry-wise product, and $\mathbf{1} \in \mathbb{R}^N$ is the vector of all ones. We call this conditional coupling (C-coupling).

其中 $X_1 \sim q(X_1)$，$I \in \{0,1\}^N$ 是标识条件位置的随机变量，$\odot$ 表示逐元素乘积，$\mathbf{1} \in \mathbb{R}^N$ 是全一向量。此方式称为条件耦合（C-coupling）。

2.3 Probability paths

2.3 概率路径

S012 · Original / 原文 · PDF p.3 · S2.SS3.p1.1

We follow the Flow Matching approach *(Lipman et al., 2022; Liu et al., 2022; Albergo and Vanden-Eijnden, 2022)* that uses a predefined *probability path* p_t interpolating p and q, i.e.,

我们遵循流匹配方法（Lipman 等，2022；Liu 等，2022；Albergo 与 Vanden-Eijnden，2022），使用一条预定义的概率路径 p_t 在 p 与 q 之间插值，即满足下列端点条件。

$$p_0 = p \quad \text{and} \quad p_1 = q \tag{6}$$

S013 · Original / 原文 · PDF p.3 · S2.SS3.p1.2

to train the generative model taking a source sample $X_0 \sim p$ to a target sample $X_1 \sim q$. We use arbitrary coupling of source and target *(Pooladian et al., 2023; Tong et al., 2023)*, $\pi(x_0, x_1)$, and the symmetric Flow Matching path *(Albergo and Vanden-Eijnden, 2022)* to define the marginal probability path,

由此训练把源样本 $X_0 \sim p$ 变为目标样本 $X_1 \sim q$ 的生成模型。我们采用任意源—目标耦合 $\pi(x_0, x_1)$（Pooladian 等，2023；Tong 等，2023），以及对称流匹配路径（Albergo 与 Vanden-Eijnden，2022），定义边缘概率路径如下。

$$p_t(x) = \sum_{x_0, x_1 \in \mathcal{D}} p_t(x|x_0, x_1)\pi(x_0, x_1), \text{ where } p_t(x|x_0, x_1) = \prod_{i=1}^{N} p_t(x^i|x_0, x_1), \tag{7}$$

S014 · Original / 原文 · PDF p.3 · S2.SS3.p1.3

and $p_t(x^i|x_0, x_1)$ is a time-dependent probability on the space of tokens $[d]$ conditioned on the pair x_0, x_1, and satisfying $p_0(x^i|x_0, x_1) = \delta_{x_0}(x^i)$ and $p_1(x^i|x_0, x_1) = \delta_{x_1}(x^i)$. If the conditional path $p_t(x^i|x_0, x_1)$ satisfies these boundary conditions then the marginal path $p_t(x)$ satisfies (6).

这里 $p_t(x^i|x_0, x_1)$ 是给定样本对 x_0, x_1 后、词元空间 [d] 上随时间变化的概率，满足 $p_0(x^i|x_0, x_1) = \delta_{x_0}(x^i)$ 与 $p_1(x^i|x_0, x_1) = \delta_{x_1}(x^i)$。若条件路径满足这些边界条件，则边缘路径 $p_t(x)$ 满足式（6）。

S015 · Original / 原文 · PDF p.3 · S2.SS3.p2.1

In developing the framework, we would like to consider as general as possible set of probability paths that are also tractable to learn within the Flow Matching framework. We consider conditional probability paths as a convex sum of m conditional probabilities $w^j(x^i|x_0, x_1)$, i.e.,

构建框架时，我们希望覆盖尽可能一般、同时又能在流匹配框架内切实学习的概率路径。考虑把条件概率路径写成 m 个条件概率 $w^j(x^i|x_0, x_1)$ 的凸组合，即下式。

$$p_t(x^i|x_0, x_1) = \sum_{j=1}^{m} \kappa_t^{i,j} w^j(x^i|x_0, x_1), \tag{8}$$

S016 · Original / 原文 · PDF p.3 · S2.SS3.p2.2

where $\sum_j \kappa_t^{i,j} = 1$ and $\kappa_t^{i,j} \geq 0$ are collectively called the *scheduler*. Note that the scheduler can be defined independently for each location in the sequence $i \in [N]$ or uniformly for all tokens, $\kappa_t^{i,j} = \kappa_t^j$.

满足 $\sum_j \kappa_t^{i,j} = 1$ 与 $\kappa_t^{i,j} \geq 0$ 的这组系数统称调度器。调度器既可对序列的每个位置 $i \in [N]$ 独立定义，也可对全部词元统一设置为 $\kappa_t^{i,j} = \kappa_t^j$。

S017 · Original / 原文 · PDF p.3 · S2.SS3.p3.1

A simple yet useful instance of these conditional paths is reminiscent of the continuous Flow Matching paths formulated as convex interpolants,

下式是一个简单而有用的条件路径实例，它类似于连续流匹配中用凸插值构造的路径。

$$p_t(x^i|x_0,x_1) = (1-\kappa_t)\delta_{x_0}(x^i) + \kappa_t\delta_{x_1}(x^i), \quad (9)$$

S018 · Original / 原文 · PDF p.3 · S2.SS3.p3.2

where the scheduler κ_t satisfies $\kappa_0 = 0$, $\kappa_1 = 1$, and monotonically increasing in t. Another interesting instantiation of (8) is adding uniform noise with some probability depending on t,

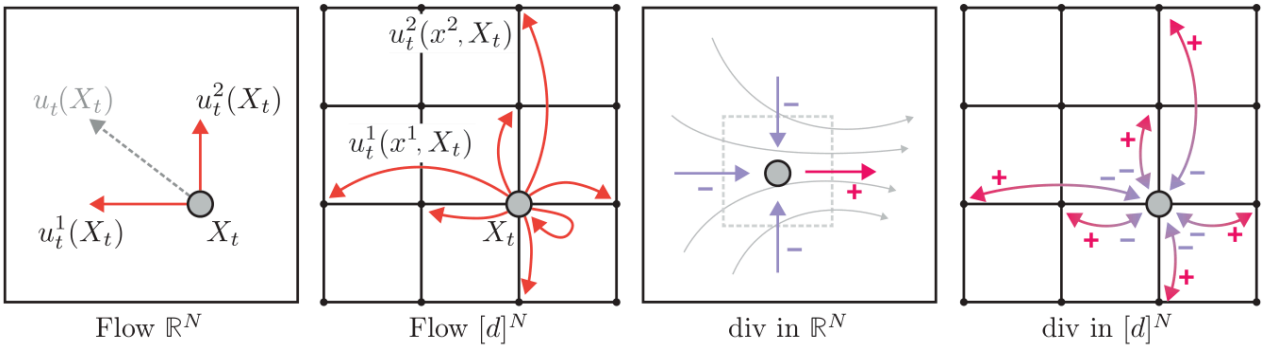
调度器 κ_t 满足 $\kappa_0=0$、$\kappa_1=1$，且关于 t 单调递增。式（8）的另一个有趣实例，是以随时间变化的概率添加均匀噪声。

$$p_t(x^i|x_0,x_1) = \kappa_t^1\delta_{x_1}(x^i) + \kappa_t^2 p_u(x^i) + \kappa_t^3\delta_{x_0}(x^i), \quad (10)$$

S019 · Original / 原文 · PDF p.3 · S2.SS3.p3.3

where $\kappa_0^1 = 0$, $\kappa_1^1 = 1$, $\kappa_0^2 = \kappa_1^2 = 0$ (remembering that $\sum_j \kappa_t^{i,j} = 1$ and $\kappa_t^{i,j} \geq 0$).

其中 $\kappa_0^1 = 0$、$\kappa_1^1 = 1$、$\kappa_0^2 = \kappa_1^2 = 0$；同时仍须满足 $\sum_j \kappa_t^{i,j} = 1$、$\kappa_t^{i,j} \geq 0$。



源图 / 表 / 代码：英文 PDF 第 4 页 · S2.F2

Figure 2: Discrete flow in $\mathcal{D} = [d]^N$ with $d = 4, N = 2$ (middle-left) versus continuous flow in $\mathbb{R}^N$, $N = 2$ (left). The rate of change of probability of a state (gray disk) is given by the divergence operator shown in the continuous case (middle right) and the discrete case (right).

图 2：$D=[d]^N$ 中 d=4、N=2 的离散流（中左）与 N=2 的连续流（左）。灰色圆盘状态的概率变化率由散度算子给出；中右、右分别为连续与离散散度。

2.4 Generating Probability Velocities

2.4 生成概率速度

S020 · Original / 原文 · PDF p.4 · S2.SS4.p1.1

Continuous generating velocity. Sampling in continuous Flow Matching is performed by updating the current (continuous) sample $X_t \in \mathbb{R}^N$, $t \in [0,1)$, according to a learned *generating velocity field* $u_t^i(X_t)$, $i \in [N]$. Euler sampling follows the (deterministic) rule

连续生成速度。连续流匹配依据学得的生成速度场 $u_t^i(X_t)$（$i \in [N]$），更新当前连续样本 $X_t \in \mathbb{R}^N$，其中 $t \in [0,1)$。Euler 采样遵循下列确定性更新规则。

$$X_{t+h}^i = X_t^i + h u_t^i(X_t), \quad (11)$$

S021 · Original / 原文 · PDF p.4 · S2.SS4.p1.2

where $h > 0$ is a user-defined time step. Note that (11) is updating separately each of the sample coordinates, X_t^i, $i \in [N]$, see e.g., Figure 2, left. The velocity $u_t^i(X_t)$ can be either directly modeled with a neural network, or parameterized via the *denoiser* (a.k.a. x-prediction) or *noise-prediction* (a.k.a. ε-prediction), see left column in Table 1. If, for all $t \in [0, 1)$, starting at $X_t \sim p_t$ and sampling with (11) provides $X_{t+h} \sim p_{t+h} + o(h)$ 1 The $o(h^\ell)$ notation means a function going to zero faster than h^ℓ as $h \to 0$, i.e., $\frac{o(h^\ell)}{h^\ell} \xrightarrow{h \to 0} 0$. then we say that u_t *generates* p_t.

其中 h>0 是用户指定的时间步长。注意式（11）分别更新样本的每个坐标 X_t^i，例如图 2 左图。速度 $u_t^i(X_t)$ 可以直接由神经网络建模，也可以通过去噪器（x 预测）或噪声预测器（ε 预测）参数化，见表 1 左列。如果对任意 $t \in [0, 1)$，从 $X_t \sim p_t$ 出发按式（11）采样所得 $X_{t+h} \sim p_{t+h} + o(h)$，就称 u_t 生成 p_t。注 1：$o(h^\ell)$ 表示当 h→0 时比 h^ℓ 更快趋于零的函数，即 $o(h^\ell)/h^\ell \to 0$。

Generating probability velocity.

生成概率速度

S022 · Original / 原文 · PDF p.4 · S2.SS4.SSS0.Px1.p1.1

For defining Flow Matching in the discrete setting, we follow *Campbell et al. (2024)* and consider a Continuous-Time discrete Markov Chain (CTMC) paradigm, namely the sample X_t is jumping between states in $\mathcal{D}$, depending on a continuous time value $t \in [0, 1]$. Similar to the continuous Flow Matching setting described above, we focus on a model that predicts the rate of probability change of the current sample X_t in each of its N tokens, see Figure 2, middle-left. Then, each token of the sample $X_t \sim p_t$ is updated independently by

为在离散情形下定义流匹配，我们遵循 Campbell 等（2024），采用连续时间离散马尔可夫链（CTMC）范式：样本 X_t 随连续时间 $t \in [0, 1]$ 在 D 的状态之间跳跃。与上述连续流匹配相似，模型预测当前样本 X_t 各个词元的概率变化率，见图 2 中左图。随后对 $X_t \sim p_t$ 的每个词元独立更新如下。

$$X_{t+h}^i \sim \delta_{X_t^i}(\cdot) + h u_t^i(\cdot, X_t), \tag{12}$$

S023 · Original / 原文 · PDF p.4 · S2.SS4.SSS0.Px1.p1.2

where we call u_t the *probability velocity* as reminiscent of the velocity field in continuous Flow Matching, and as in the continuous case, we define:

我们把 u_t 称为概率速度，呼应连续流匹配中的速度场。类似连续情形，作如下定义。

> Definition 1. 定义 1
>
> S024 · Original / 原文 · PDF p.4 · S2.Thmtheorem1.p1.1
>
> Probability velocity u_t *generates* the probability path p_t if, for all $t \in [0, 1)$ and given a sample $X_t \sim p_t$, the sample X_{t+h} defined in (12) satisfies $X_{t+h} \sim p_{t+h} + o(h)$.
>
> 若对所有 $t \in [0, 1)$ 及给定样本 $X_t \sim p_t$，由式（12）定义的样本 X_{t+h} 满足 $X_{t+h} \sim p_{t+h} + o(h)$，则称概率速度 u_t 生成概率路径 p_t。

S025 · Original / 原文 · PDF p.4 · S2.SS4.SSS0.Px1.p2.1

Algorithm 1 formulates a basic sampling algorithm given a generating probability velocity u_t. In order for the r.h.s. of (12) to define a proper PMF for sufficiently small $h > 0$, it is necessary and sufficient that the probability velocity satisfies the conditions

算法 1 给出了使用生成概率速度 u_t 的基本采样算法。要使式（12）右侧在足够小的 h>0 下定义有效 PMF，概率速度满足以下条件是必要且充分的。

$$\sum_{x^i \in [d]} u_t^i(x^i, z) = 0, \text{ and } u_t^i(x^i, z) \geq 0 \text{ for all } i \in [N] \text{ and } x^i \neq z^i. \tag{13}$$

Algorithm 1 Flow Matching sampling.

Require: velocity u_t, sample $X \sim p$, step size $h = \frac{1}{n}$
  for $t = 0, h, 2h, \ldots, 1-h$ **do**
    $X^i \sim \delta_{X^i}(\cdot) + h u_t^i(\cdot, X)$, for $i \in [N]$ ▷ eq. 24 or 22
  end for
  return X

$x_0, x_1)$. This can also be shown to hold in the discrete
nulation in our context and notation is as follows.

源图 / 表 / 代码：英文 PDF 第 4 页 · S2.SS4.SSS0.Px1.fig1

Algorithm 1 Flow Matching sampling.

算法 1：流匹配采样。输入速度 u_t、初始样本 X~p、步长 h=1/n；对 t=0,h,2h,...,1−h，逐词元独立按 $\delta X^i + h\, u_t^i(\cdot, X)$ 抽样（式 24 或 22），循环结束返回 X。

S026 · Original / 原文 · PDF p.4 · S2.SS4.SSS0.Px1.p3.1

Now the main question is how to find a probability velocity u_t that generates the probability path defined in equations 7 and 8? A key insight in Flow Matching *(Lipman et al., 2022)* is that u_t can be constructed as a marginalization of *conditional* probability velocities, $u_t^i(x^i, z|x_0, x_1)$, generating the corresponding conditional probability paths $p_t(x^i|x_0, x_1)$. This can also be shown to hold in the discrete CTMC setting *(Campbell et al., 2024)*, where a reformulation in our context and notation is as follows.

核心问题是：如何找到生成式（7）、（8）所定义概率路径的概率速度 u_t？流匹配的关键洞见（Lipman 等，2022）在于，可以对条件概率速度 $u_t^i(x^i, z|x_0, x_1)$ 做边缘化来构造 u_t；这些条件速度分别生成对应条件概率路径 $p_t(x^i|x_0, x_1)$。该结论在离散 CTMC 情形也成立（Campbell 等，2024），用本文符号可表述如下。

Theorem 2. 定理 2

S027 · Original / 原文 · PDF p.5 · S2.Thmtheorem2.p1.1

Given a conditional probability velocity $u_t^i(x^i, z|x_0, x_1)$ generating a conditional probability path $p_t(x|x_0, x_1)$, the marginal velocity defined by

给定生成条件概率路径 $p_t(x|x_0, x_1)$ 的条件概率速度 $u_t^i(x^i, z|x_0, x_1)$，按下式定义的边缘速度：

$$u_t^i(x^i, z) = \sum_{x_0, x_1 \in \mathcal{D}} u_t^i(x^i, z|x_0, x_1) p_t(x_0, x_1|z) \tag{14}$$

S028 · Original / 原文 · PDF p.5 · S2.Thmtheorem2.p1.2

generates the marginal probability path $p_t(x)$, where by Bayes' rule

生成边缘概率路径 $p_t(x)$，其中由 Bayes 公式得到下式。

$$p_t(x_0, x_1|z) = \frac{p_t(z|x_0, x_1)\pi(x_0, x_1)}{p_t(z)}. \tag{15}$$

S029 · Original / 原文 · PDF p.5 · S2.SS4.SSS0.Px1.p4.1

For completeness we provide a simple proof of this theorem in Section E.2. The proof, similar to the continuous Flow Matching case, shows that u_t and p_t satisfy the (discrete version of the) Continuity Equation.

为使论述完整，附录 E.2 给出此定理的简短证明。与连续流匹配类似，证明表明 u_t 和 p_t 满足离散版本的连续性方程。

The Continuity Equation.

连续性方程

S030 · Original / 原文 · PDF p.5 · S2.SS4.SSS0.Px2.p1.2

To provide the mathematical tool for showing that a probability velocity u_t does indeed generate the probability path p_t, and also to further highlight the similarities to the continuous case, we next formulate the *Kolmogorov Equations*, which describe the state probability rate $\dot{p}_t(x)$, $x \in \mathcal{D}$, in CTMC as a Continuity Equation (CE). The Continuity Equation, similarly to Kolmogorov Equations, describes $\dot{p}_t(x)$, $x \in \mathbb{R}^N$ in the *continuous case*, and is formulated as the Partial Differential Equation (PDE)

为提供检验概率速度 u_t 是否确实生成概率路径 p_t 的数学工具，并进一步揭示与连续情形的相似性，我们将描述 CTMC 中状态概率变化率 $\dot{p}_t(x)$（x∈D）的 Kolmogorov 方程改写为连续性方程（CE）。连续情形中，连续性方程同样描述 $\dot{p}_t(x)$（$x \in \mathbb{R}^N$），其偏微分方程（PDE）形式如下。

$$\dot{p}_t(x) + \mathrm{div}_x(p_t u_t) = 0, \tag{16}$$

S031 · Original / 原文 · PDF p.5 · S2.SS4.SSS0.Px2.p1.3

where the divergence operator $\mathrm{div}_x(v)$ applied to a vector field $v : \mathbb{R}^N \to \mathbb{R}^N$ is defined by

其中，对向量场 $v : \mathbb{R}^N \to \mathbb{R}^N$ 的散度算子定义如下。

$$\mathrm{div}_x(v) = \sum_{i=1}^{N} \partial_{x^i} v^i(x), \tag{17}$$

S032 · Original / 原文 · PDF p.5 · S2.SS4.SSS0.Px2.p1.4

and intuitively means the total flux leaving x, see Figure 2 (middle-right). This gives an intuitive explanation to the Continuity Equation: the rate of the probability $\dot{p}_t(x)$ of a state $x \in \mathbb{R}^N$ equals the total *incoming probability flux*, $p_t u_t$, at x. In the discrete case (CTMC) the Continuity Equation ((16)) holds as is, once the discrete divergence operator is properly defined, i.e., to measure the outgoing flux from a discrete state. In more detail, given some vector field, which in the discrete case is a scalar-valued function over pairs of states, $v : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$, the discrete divergence is

直观上，它表示离开 x 的总通量，见图 2 中右图。这给出了连续性方程的直观解释：状态 x 的概率变化率等于该处总流入概率通量 $p_t u_t$。离散情形（CTMC）只要恰当定义离散散度、使它度量离开离散状态的通量，式（16）即可原样成立。具体而言，离散向量场是定义在状态对上的标量函数 $v : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$，其离散散度为下式。

$$\mathrm{div}_x(v) = \sum_{z \in \mathcal{D}} \left[v(z, x) - v(x, z) \right], \tag{18}$$

S033 · Original / 原文 · PDF p.5 · S2.SS4.SSS0.Px2.p1.5

where $v(z, x)$ represents the flux $x \to z$ and $v(x, z)$ represent the opposite flux $z \to x$; see Figure 2, right. Now, in our case (see Figure 2, middle-left), the probability flux at a state $x \in \mathcal{D}$ involves all sequences with at most one token difference from x,

i.e., the probability flux $p_t u_t$ at x takes the form $v(x,z) = p_t(z)u_t^i(x^i,z)$ and $v(z,x) = p_t(x)u_t^i(z^i,x)$ for z and x that differ only in the i-th token, $v(x,x) = p_t(x)\sum_{i=1}^N u_t^i(x^i,x)$, and $v(x,z) = 0$ for all other $(z,x) \in \mathcal{D}\times\mathcal{D}$. A direct calculation now shows (see Section E.1):

这里 v(z,x) 表示 x→z 的通量，v(x,z) 表示反向 z→x 的通量，见图 2 右图。本文中（见图 2 中左），状态 x 处的概率通量仅涉及与 x 至多相差一个词元的序列。若 z 与 x 仅第 i 个词元不同，则 $v(x,z) = p_t(z)u_t^i(x^i,z)$、$v(z,x) = p_t(x)u_t^i(z^i,x)$；自环通量为 $v(x,x) = p_t(x)\sum_i u_t^i(x^i,x)$；其余状态对的 v(x,z) 为零。直接计算得到下式，详见附录 E.1。

$$\mathrm{div}_x(p_t u_t) = -\sum_{z\in\mathcal{D}} p_t(z)\left[\sum_{i=1}^N \delta_z(x^{\bar{i}})u_t^i(x^i,z)\right]. \tag{19}$$

S034 · Original / 原文 · PDF p.5 · S2.SS4.SSS0.Px2.p1.6

Checking that a probability velocity u_t generates a probability path p_t (in the sense of Definition 1) amounts to verifying the Continuity Equation ((16)). Indeed, using arguments from *Campbell et al. (2024)* and the discrete divergence operator, the PMF of X_{t+h} defined by sampling according to (12) is

检验概率速度 u_t 是否在定义 1 的意义下生成 p_t，归结为验证连续性方程（16）。确实，采用 Campbell 等（2024）的论证与离散散度算子，按式（12）采样的 X_{t+h} 的 PMF 如下。

$$\begin{aligned}\mathbb{E}_{X_t}\prod_{i=1}^N\left[\delta_{X_t}(x^i) + hu_t^i(x^i,X_t)\right] &= \mathbb{E}_{X_t}\left[\delta_{X_t}(x) + h\sum_{i=1}^N \delta_{X_t}(x^{\bar{i}})u_t^i(x^i,X_t)\right] + o(h)\\ &= p_t(x) - h\mathrm{div}_x(p_t u_t) + o(h) \overset{(16)}{=} p_t(x) + h\dot{p}_t(x) + o(h) = p_{t+h}(x) + o(h),\end{aligned} \tag{20}$$

S035 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p1.7

where we assume $X_t \sim p_t$, the first equality uses the identity $\prod_i\left[a^i + hb^i\right] = \prod_i a^i + h\sum_i(\prod_{j\neq i}a^j)b^i + o(h)$, the second equality uses (19), and the previous-to-last equality uses the Continuity Equation ((16)). This shows that if the Continuity Equation holds then u_t generates p_t in the sense of Definition 1.

这里假设 $X_t \sim p_t$。第一个等号使用恒等式 $\prod_i(a^i + hb^i) = \prod_i a^i + h\sum_i(\prod_{j\neq i}a^j)b^i + o(h)$；第二个等号使用式（19）；倒数第二个等号使用连续性方程（16）。由此，连续性方程成立就意味着 u_t 在定义 1 的意义下生成 p_t。

S036 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p2.1

Conditional and marginal generating velocities. We provide the probability velocities generating the conditional probability paths $p_t(x|x_0,x_1)$ defined in equations 7 and 8. Then, using the marginalization formula in (14) we end up with a closed-form marginal velocity for the probability paths $p_t(x)$. In Section E.3 we show

条件与边缘生成速度。我们先给出生成式（7）、（8）所定义条件概率路径 $p_t(x|x_0,x_1)$ 的概率速度，再用式（14）的边缘化公式，得到生成边缘路径 $p_t(x)$ 的闭式边缘速度。附录 E.3 证明如下结论。

Theorem 3 (Probability velocity of conditional paths). 定理 3（条件路径的概率速度）

S037 · Original / 原文 · PDF p.6 · S2.Thmtheorem3.p1.1

A generating probability velocity for the conditional paths $p_t(x|x_0,x_1)$ defined in equations 7 and 8 is

生成式（7）、（8）所定义条件概率路径 $p_t(x|x_0,x_1)$ 的一个概率速度如下。

$$u_t^i(x^i,z|x_0,x_1) = \sum_{j=1}^m a_t^{i,j}w^j(x^i|x_0,x_1) + b_t^i\delta_z(x^i), \tag{21}$$

S038 · Original / 原文 · PDF p.6 · S2.Thmtheorem3.p1.2

with $a_t^{i,j} = \dot{\kappa}_t^{i,j} - \kappa_t^{i,j}\dot{\kappa}_t^{i,\ell}/\kappa_t^{i,\ell}$, and $b_t^i = \dot{\kappa}_t^{i,\ell}/\kappa_t^{i,\ell}$ where $\ell = \arg\min_{j\in[m]}\left[\dot{\kappa}_t^{i,j}/\kappa_t^{i,j}\right]$.

其中 $a_t^{i,j} = \dot\kappa_t^{i,j} - \kappa_t^{i,j}\dot\kappa_t^{i,\ell}/\kappa_t^{i,\ell}$，$b_t^i = \dot\kappa_t^{i,\ell}/\kappa_t^{i,\ell}$，且 $\ell = \arg\min_{j\in[m]}(\dot\kappa_t^{i,j}/\kappa_t^{i,j})$。

S039 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p3.1

Now, computing the marginal probability velocity using (14) applied to the conditional probability velocity in (21) gives

把式（14）作用于式（21）的条件概率速度，便得到下列边缘概率速度。

$$u_t^i(x^i, z) = \sum_{j=1}^{m} a_t^{i,j}\hat w_t^j(x^i, z) + b_t^{i,j}\delta_z(x^i), \tag{22}$$

S040 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p3.2

where the posteriors $\hat w_t^j$ of w^j (that are later shown to be tractable to learn) are defined by

其中 w^j 的后验 $\hat w_t^j$ 定义如下；后文将证明这些后验可以切实学习。

$$\hat w_t^j(x^i, z) = \sum_{x_0, x_1 \in \mathcal{D}} w^j(x^i|x_0, x_1)p_t(x_0, x_1|z), \tag{23}$$

S041 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p3.3

where $p_t(x_0, x_1|z)$ (defined in (15)) is the posterior probability of x_0, x_1 conditioned on the current state $X_t = z$. A useful instantiation of the general velocity in (22) is when considering the path family in (9), for which $w^1(x^i|x_0, x_1) = \delta_{x_1}(x^i)$, $w^2(x^i|x_0, x_1) = \delta_{x_0}(x^i)$, $\kappa_t^{i,1} = \kappa_t$, $\kappa_t^{i,2} = 1 - \kappa_t$, $\dot\kappa_t \geq 0$ (i.e., monotonically non-decreasing in t) and in this case (22) reads as

其中式（15）定义的 $p_t(x_0, x_1|z)$ 是给定当前状态 X_t=z 的端点后验。对于式（9）的路径族，一般速度式（22）有一个有用的特例：$w^1 = \delta_{x_1}$、$w^2 = \delta_{x_0}$、$\kappa_t^{i,1} = \kappa_t$、$\kappa_t^{i,2} = 1 - \kappa_t$，并且 $\dot\kappa_t \geq 0$，即随 t 单调不减。此时式（22）化为下式。

S042 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p3.4

$$u_t^i(x^i, z) = \frac{\dot\kappa_t}{1 - \kappa_t}\left[p_{1|t}(x^i|z) - \delta_z(x^i)\right] \tag{24}$$

式（24）：概率速度等于调度危险率 $\dot\kappa_t/(1-\kappa_t)$ 乘以概率去噪器与当前状态 delta 分布的差。原式如下保留。

S043 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px2.p3.5

where we use the notation $p_{1|t}(x^i|z) = \sum_{x_0,x_1}\delta_{x_1}(x^i)p_t(x_0, x_1|z)$ for the *probability denoiser*.

这里概率去噪器记为 $p_{1|t}(x^i|z) = \sum_{x_0,x_1}\delta_{x_1}(x^i)p_t(x_0, x_1|z)$。

	Continuous Flow Matching	**Discrete** Flow Matching
Marginal prob.	$p_t(x) = \sum_{x_0,x_1} \prod_{i=1}^N p_t(x^i\|x_0,x_1)\pi(x_0,x_1)$	
Conditional prob.	$p_t(x^i\|x_0,x_1) = \delta_{\kappa_t x_1+(1-\kappa_t)x_0}(x^i)$	$p_t(x^i\|x_0,x_1) = \kappa_t\delta_{x_1}(x^i) + (1-\kappa_t)\delta_{x_0}(x^i)$
VF-*Denoiser*	$u_t^i(X_t) = \frac{\dot\kappa_t}{1-\kappa_t}\left[\hat x^i_{1\|t}(X_t) - X_t^i\right]$	$u_t^i(x^i,X_t) = \frac{\dot\kappa_t}{1-\kappa_t}\left[p_{1\|t}(x^i\|X_t) - \delta_{X_t}(x^i)\right]$
VF-*Noise-pred*	$u_t^i(X_t) = \frac{\dot\kappa_t}{\kappa_t}\left[X_t^i - \hat x^i_{0\|t}(X_t)\right]$	$u_t^i(x^i,X_t) = \frac{\dot\kappa_t}{\kappa_t}\left[\delta_{X_t}(x^i) - p_{0\|t}(x^i\|X_t)\right]$

源图 / 表 / 代码：英文 PDF 第 6 页 · S2.T1

Table 1: Generating (marginal) velocity fields have identical form for the continuous and discrete Flow Matching when using denoiser/noise-prediction parameterization; $\hat x_{1|t}(z) = \mathbb{E}_{X_1\sim p_t(\cdot|z)}X_1$ is the standard continuous denoiser (a.k.a. x-prediction) and $\hat x_{0|t}(z) = \mathbb{E}_{X_0\sim p_t(\cdot|z)}X_0$ is the standard noise-prediction (a.k.a. ϵ-prediction).

表 1：用去噪器或噪声预测器参数化时，连续与离散流匹配的生成（边缘）速度具有相同形式。蓝色 $\hat{x}_{1}|_t(z)$ 是标准连续去噪器 $\mathbb{E}[X_1|z]$（x 预测），红色 $\hat{x}_{0}|_t(z)$ 是标准噪声预测器 $\mathbb{E}[X_0|z]$（ε 预测）。

Sampling backward in time.

反向时间采样

S044 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px3.p1.1

We can also sample *backwards in time* by following the sampling rule $X^i_{t-h} \sim \delta_{X_t^i}(\cdot) - hu_t^i(\cdot, X_t)$. In this case $-u_t^i(x^i, z)$ should satisfy (13). A (backward-time) generating probability velocity can then be achieved from (22) with the simple change to the coefficients $a_t^{i,j}$ and $b_t^{i,j}$, see Section E.4. For p_t defined with (9) the generating velocity is

也可以按 $X^i_{t-h} \sim \delta_{X_t^i}(\cdot) - hu_t^i(\cdot, X_t)$ 反向采样。此时 $-u_t^i(x^i, z)$ 应满足式（13）。仅需改变式（22）中的系数 $a_t^{i,j}$、$b_t^{i,j}$，即可得到反向时间生成概率速度，详见附录 E.4。对于用式（9）定义的 p_t，其生成速度如下。

S045 · Original / 原文 · PDF p.6 · S2.SS4.SSS0.Px3.p1.2

$$u_t^i(x^i, z) = \frac{\dot\kappa_t}{\kappa_t}\left[\delta_z(x^i) - p_{0|t}(x^i|z)\right], \tag{25}$$

式（25）：反向参数化使用 $\dot\kappa_t/\kappa_t$ 乘以当前 delta 分布与概率噪声预测器的差。原式如下保留。

S046 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px3.p1.3

where in this case $p_{0|t}(x^i|z) = \sum_{x_0,x_1\in\mathcal{D}}\delta_{x_0}(x^i)p_t(x_0,x_1|z)$ is the *probability noise-prediction*.

这里 $p_{0|t}(x^i|z) = \sum_{x_0,x_1\in\mathcal{D}}\delta_{x_0}(x^i)p_t(x_0,x_1|z)$ 是概率噪声预测器。

S047 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px3.p2.1

Remarkably, the generating velocity fields in 24 and 25 take the *exact same form* as the generating (a.k.a. marginal) velocity fields in *continuous* flow matching when parameterized via the denoiser or noise-prediction parameterizations and using the same schedulers, see Table 1 and Section E.9 for explanation of the continuous case. In Section E.4 we provide the backward-time version of Theorem 3.

值得注意的是，式（24）、（25）与连续流匹配中用去噪器或噪声预测器参数化、且采用相同调度器时的生成（边缘）速度场具有完全相同的形式。连续情形的解释见表 1 与附录 E.9。附录 E.4 还给出了定理 3 的反向时间版本。

Corrector sampling.

校正采样

S048 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px4.p1.1

Combining the forward-time $\hat u_t$ ((24)) and backward-time $\check u_t$ ((25)), i.e.,

将前向时间速度 $\hat{u}_t$（式 24）与反向时间速度 $\check{u}_t$（式 25）作下列组合。

$$\bar{u}_t^i(x^i, z) = \alpha_t \hat{u}_t^i(x^i, z) - \beta_t \check{u}_t^i(x^i, z), \tag{26}$$

S049 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px4.p1.2

provides a valid forward-time probability velocity field (i.e., satisfies (13)) for $t \in (0,1)$ as long as $\alpha_t, \beta_t > 0$. This velocity field can be used for two types of corrector sampling: (i) When $\alpha_t - \beta_t = 1$ sampling with $\bar{u}_t$ leads to *corrector sampling* where intuitively each step moves $1 + \beta_t$ forward in time and $-\beta_t$ backwards, which allows reintroducing noise into the sampling process; and (ii) when $\alpha_t - \beta_t = 0$ sampling with $\bar{u}_t$ when fixing $t \in (0,1)$ leads to *corrector iterations* where limit samples distribute according to p_t. In Section E.6 we prove:

只要 $\alpha_t, \beta_t > 0$，就可在 $t \in (0,1)$ 得到有效前向概率速度，即满足式（13）。该速度有两种校正用法：（i）当 $\alpha_t - \beta_t = 1$ 时，每一步直观上向前推进 $1+\beta_t$、向后退回 $-\beta_t$，因此可把噪声重新引入采样过程；（ii）当 $\alpha_t - \beta_t = 0$ 时，固定 $t \in (0,1)$ 反复采样可形成校正迭代，其极限样本服从 p_t。附录 E.6 证明如下结论。

> **Theorem 4.** 定理 4
>
> S050 · Original / 原文 · PDF p.7 · S2.Thmtheorem4.p1.1
>
> For perfectly trained posteriors and $\alpha_t, \beta_t > 0$, $t \in (0,1)$, $\bar{u}_t$ in (26) is a probability velocity, i.e., satisfies (13), and: (i) For $\alpha_t - \beta_t = 1$, $\bar{u}_t$ provides a probability velocity generating p_t; (ii) For $\alpha_t - \beta_t = 0$, repeatedly sampling with $\bar{u}_t$ at fixed $t \in (0,1)$ and sufficiently small h is guaranteed to converge to a sample from p_t.
>
> 对完全训练好的后验，若 $\alpha_t, \beta_t > 0$、$t \in (0,1)$，则式（26）的 $\bar{u}_t$ 是满足式（13）的概率速度；（i）$\alpha_t - \beta_t = 1$ 时，它生成 p_t；（ii）$\alpha_t - \beta_t = 0$ 时，固定 t 并取足够小的 h，反复按此速度采样，保证收敛到服从 p_t 的样本。

S051 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px4.p2.1

One simplification to (26) can be done in the case of paths constructed with conditional as in (9), independent coupling $\pi(x_0, x_1) = p(x_0)q(x_1)$, and i.i.d. source $p(x_0) = \prod_{i=1}^N p(x_0^i)$, e.g., $p(x_0^i)$ is uniform over $[d]$ or $\delta_m(x_0^i)$. In this case, the backward-time formula in (25) take an equivalent simpler form

若路径的条件分布为式（9）、耦合独立 $\pi(x_0, x_1) = p(x_0)q(x_1)$，且源分布独立同分布 $p(x_0) = \prod_i p(x_0^i)$，则式（26）可简化。例如每个源词元可以服从 [d] 上均匀分布或 mask 的 delta 分布。此时式（25）的反向速度有下列等价简式。

$$\check{u}_t^i(x^i, z) = \frac{\dot{\kappa}_t}{\kappa_t}\left[\delta_z(x^i) - p(x^i)\right], \tag{27}$$

S052 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px4.p2.2

which does not require estimation of the posterior $p_{0|t}$. See Section E.5 for the derivation.

这一简式不需要估计后验 $p_{0|t}$，推导见附录 E.5。

Training.

训练

S053 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px5.p1.1

Equation 22 shows that for generating samples from a probabilty path $p_t(x)$ we require the posteriors $\hat{w}_t^j(x^i|X_t)$. Training such posteriors can be done by minimizing the loss

式（22）表明，沿概率路径 $p_t(x)$ 生成样本需要后验 $\hat{w}_t^j(x^i|X_t)$。这些后验可通过最小化下列损失训练。

$$\mathcal{L}(\theta) = -\sum_{j\in[m],i\in[N]} \mathbb{E}_{t,(X_0,X_1),X_t,Y_j^i} \log \hat{w}_t^j(Y_j^i|X_t;\theta), \tag{28}$$

S054 · Original / 原文 · PDF p.7 · S2.SS4.SSS0.Px5.p1.2

where t is sampled according to some distribution in $[0,1]$ (we used uniform), $(X_0,X_1)\sim\pi(X_0,X_1)$, $X_t\sim p_t(X_t|X_0,X_1)$, and $Y_j^i\sim w^j(Y_j^i|X_0,X_1)$; $\theta\in\mathbb{R}^p$ denotes the learnable parameters. In the common case we use in this paper of learning a single posterior, i.e., the probability denoiser $p_{1|t}$, the loss takes the form $\mathcal{L}(\theta) = -\sum_{i\in[N]}\mathbb{E}_{t,(X_0,X_1),X_t}\log p_{1|t}(X_1^i|X_t)$. In Section E.7 we prove:

其中 t 从 [0,1] 上某分布采样（本文用均匀分布），$(X_0,X_1)\sim\pi$、$X_t\sim p_t(X_t|X_0,X_1)$，且 $Y_j^i\sim w^j(Y_j^i|X_0,X_1)$；$\theta\in\mathbb{R}^p$ 是可学习参数。本文常用情形只学习单一后验，即概率去噪器 $p_{1|t}$，此时损失为 $\mathcal{L}(\theta) = -\sum_i \mathbb{E}_{t,(X_0,X_1),X_t}\log p_{1|t}(X_1^i|X_t)$。附录 E.7 证明如下结论。

> **Proposition 5.** 命题 5
>
> S055 · Original / 原文 · PDF p.7 · S2.Thmtheorem5.p1.1
>
> The minimizer of $\mathcal{L}$ ((28)) is $\hat{w}_t^j(x^i|X_t)$ ((23)).
>
> 损失 L（式 28）的极小化解是式（23）的 $\hat{w}_t^j(x^i|X_t)$。

Method	NFE	Llama-2↓	Llama-3↓	GPT2↓	Entropy
Data	-	7.0	9.4	14.7	7.7
Autoregressive	1024	31.4	54.8	45.3	7.1
Savinov et al. (2021)	200	29.5	45.1	34.7	5.2
Austin et al. (2021a)	1000	697.6	768.8	837.8	7.6
Han et al. (2022)	>10000	73.3	203.1	99.2	4.8
Lou et al. (2023)	256 / 512 / 1024	38.6 / 33.7 / 27.2	69.2 / 58.6 / 43.9	64.3 / 53.4 / 40.5	7.8 / 7.7 / 7.6
Campbell et al. (2024)	256 / 512 / 1024	38.5 / 33.5 / 28.7	69.0 / 56.5 / 46.5	65.2 / 53.3 / 43.0	7.8 / 7.7 / 7.6
FM (equation 9)	256 / 512 / 1024	34.2 / 30.0 / 22.5	58.5 / 48.8 / 33.8	54.2 / 43.5 / 29.3	7.7 / 7.6 / 7.2
FM (equation 10)	256 / 512 / 1024	30.0 / 27.5 / 22.3	48.2 / 43.5 / 31.9	47.7 / 41.8 / 28.1	7.6 / 7.5 / 7.1

源图 / 表 / 代码：英文 PDF 第 8 页 · S2.T2

Table 2: Generative perplexity on unconditional text generation compared to prior work. All models are sampled without the use of temperature or corrector steps. Double precision sampling results are reported in Table 5.

表 2：无条件文本生成的生成困惑度比较；所有模型均不使用温度或校正步骤。双精度采样结果见表 5。

表头：Method = 方法，Data = 数据，NFE = 函数评估次数，Entropy = 熵，Pass@k = k 次候选内至少一项通过的比例；向下箭头为越低越好，向上箭头为越高越好。Autoregressive = 自回归，Oracle length = 给定真实解答长度。数值按原表保留，斜杠分隔值不合并。

3 Related work

3 相关工作

S056 · Original / 原文 · PDF p.7 · S3.p1.1

In the section we cover the most related work to ours; in Appendix A we cover other related work.

本节讨论与本文最相关的研究，其他相关工作见附录 A。

S057 · Original / 原文 · PDF p.7 · S3.p2.1

Discrete Flows *(Campbell et al., 2024)* is probably the most related work to ours. We build upon their CTMC framework and offer the following generalizations and simplifications: We consider arbitrary couplings (X_0, X_1), and offer a novel and rather general family of probability paths ((8)) for which we provide the generating probability velocities in a unified closed-form formula (equations 22-25). These in particular recreate the same formulas as the continuous Flow Matching counterpart (Table 1). We furthermore develop a general corrector velocity ((26)) that unifies both corrector iterations *(Song et al., 2020; Campbell et al., 2022)* and stochastic sampling of *Campbell et al. (2024)*. We show that particular choices of noise schedulers κ_t ($\kappa_t = t$ reproduces *Campbell et al. (2024)*) and corrector schedulers provide a boost in results. Lastly, we opted for the term *probability velocity* for $u_t^i(x^i, X_t)$ as it is not precisely a rate matrix in the state space $\mathcal{D} \times \mathcal{D}$ used in CTMC since $u_t^i(x^i, z)$ for all $i \in [N]$ define multiple self-edges $z \to z$.

离散流（Campbell 等，2024）或许是最接近本文的工作。我们在其 CTMC 框架上做出如下推广与简化：考虑任意端点耦合，提出相当一般的新概率路径族（式 8），并给出统一闭式生成概率速度（式 22—25），特别地重现了连续流匹配的同型公式（表 1）。一般校正速度（式 26）还统一了校正迭代（Song 等，2020；Campbell 等，2022）与 Campbell 等（2024）的随机采样。特定噪声调度器 κ_t 以及校正调度器能改善结果；κ_t=t 重现 Campbell 等的方法。我们采用“概率速度”而非“速率矩阵”，因为在 CTMC 的 D×D 状态空间里，每个 i 定义的 $u_t^i(x^i, z)$ 会产生多个 z→z 自环，因此它并不严格等于通常的速率矩阵。

S058 · Original / 原文 · PDF p.8 · S3.p3.1

Masked modeling *(Ghazvininejad et al., 2019; Chang et al., 2022)*. In case of a masked model, i.e., when the source distribution is $p(x) = \delta_m(x)$, we achieve an interesting connection with MaskGit showing it is actually an instance of Discrete Flow Matching with a small yet crucial change to its sampling algorithm. First, in Section E.8 we prove that in the masked setting, the probability denoiser $p_{1|t}$ is *time-independent*:

掩码建模（Ghazvininejad 等，2019；Chang 等，2022）。当源分布为全掩码 $p(x) = \delta_M(x)$ 时，可发现与 MaskGIT 的联系：对其采样算法做一个虽小但关键的修改，它便成为离散流匹配的实例。首先，附录 E.8 证明掩码情形的概率去噪器与时间无关。

> **Proposition 6.** 命题 6
>
> S059 · Original / 原文 · PDF p.8 · S3.Thmtheorem1.p1.1
>
> For paths defined by equations 7 and 9 with source $p(x) = \delta_m(x)$ the posterior $p_t(x_0, x_1|z) = p(x_0, x_1|z)$ is time-independent. Consequently, the probability denoiser $p_{1|t}(x^i|z) = p_1(x^i|z)$ is also time-independent.
>
> 对式（7）、（9）定义、源分布为 $p(x) = \delta_M(x)$ 的路径，后验 $p_t(x_0, x_1|z) = p(x_0, x_1|z)$ 与时间无关。因此概率去噪器 $p_{1|t}(x^i|z) = p_1(x^i|z)$ 也与时间无关。

S060 · Original / 原文 · PDF p.8 · S3.p4.1

This shows that the probability denoiser can be learned with no time dependence, similar to the unmasking probabilities in MaskGit. During sampling however, there are two main differences between our sampling and MaskGit sampling. First, unmasking of tokens in our algorithm is done according to the probability $\delta_{X_t}(x^i) + hu_t^i(x^i, X_t)$ *independently* for each token x^i, $i \in [N]$. This procedure is justified as it samples from the correct probability asymptotically via the derivation of the Continuity Equation 20. This is in contrast to MaskGit that prioritizes the token to be unmasked according to some *confidence*. In the experiments section we show that MaskGit's prioritization, although has some benefit in the very low NFE regime, is actually introducing a strong bias in the sampling procedure and leads to inferior overall results. Secondly, using corrector sampling allows for reintroducing masks to already unmasked tokens in a way that is still guaranteed to produce samples from p_t, see Theorem 4; we find this to have a significant positive effect on the generation quality.

因此可像 MaskGIT 的去掩码概率一样，学习不显含时间的概率去噪器。但两者采样有两个主要区别。第一，本文按 $\delta_{X_t}(x^i) + hu_t^i(x^i, X_t)$ 对各词元独立去掩码；连续性方程（20）的推导保证其渐近采样正确。MaskGIT 则依据某种置信度优先选择去掩码的词元。实验表明，虽然这种优先策略在极低 NFE 时有好处，却会引入显著采样偏差，使

整体结果较差。第二，校正采样可以重新掩码已解开的词元，同时仍保证样本服从 p_t（定理 4）；我们发现这明显提升生成质量。

S061 · Original / 原文 · PDF p.8 · S3.p5.1

Discrete diffusion. D3PM *(Austin et al., 2021a)* and Argmax flows *(Hoogeboom et al., 2021)* introduced diffusion in discrete spaces by proposing a corruption process for categorical data. A later work by *Campbell et al. (2022)* introduced discrete diffusion models with continuous time, and *Lou et al. (2023)* proposed learning probability ratios, extending score matching *(Song and Ermon, 2019)* to discrete spaces.

离散扩散。D3PM（Austin 等，2021a）与 Argmax flows（Hoogeboom 等，2021）通过为类别数据设计损坏过程，在离散空间引入扩散。Campbell 等（2022）随后提出连续时间离散扩散模型，Lou 等（2023）通过学习概率比率，把得分匹配（Song 与 Ermon，2019）推广至离散空间。

Method	Model Size	NFE	Llama-2↓	Llama-3↓	Entropy
Llama-3 (Reference)	8B	512	6.4	7.3	6.8
Llama-2 (Reference)	7B	512	5.3	8.3	7.1
Autoregressive	1.7B	512	14.3	22.3	7.2
Savinov et al. (2021)	1.7B	200	10.8	15.4	4.7
FM (U-coupling)	1.7B	256 / 512	10.7 / 9.5	11.2 / 10.3	6.7 / 6.7
FM (C-coupling)	1.7B	256 / 512	10.2 / 8.9	10.0 / 9.7	6.8 / 6.7

源图 / 表 / 代码：英文 PDF 第 8 页 · S3.T3

Table 3: Generative perplexity on conditional text generation.

表 3：条件文本生成的生成困惑度。

表头：Method = 方法，Data = 数据，NFE = 函数评估次数，Entropy = 熵，Pass@k = k 次候选内至少一项通过的比例；向下箭头为越低越好，向上箭头为越高越好。Autoregressive = 自回归，Oracle length = 给定真实解答长度。数值按原表保留，斜杠分隔值不合并。

Method	Data	HumanEval↑			MBPP (1-shot)↑		
		Pass@1	Pass@10	Pass@25	Pass@1	Pass@10	Pass@25
Autoregressive	Text	1.2	3.1	4.8	0.2	1.7	3.3
	Code	14.3	21.3	27.8	17.0	34.3	44.1
FM	Text	1.2	2.6	4.0	0.4	1.1	3.6
	Code	6.7	13.4	18.0	6.7	20.6	26.5
FM (Oracle length)	Code	11.6	18.3	20.6	13.1	28.4	34.2

源图 / 表 / 代码：英文 PDF 第 9 页 · S3.T4

Table 4: Execution based code generation evaluation.

表 4：基于代码执行的生成评估。

表头：Method = 方法，Data = 数据，NFE = 函数评估次数，Entropy = 熵，Pass@k = k 次候选内至少一项通过的比例；向下箭头为越低越好，向上箭头为越高越好。Autoregressive = 自回归，Oracle length = 给定真实解答长度。数值按原表保留，斜杠分隔值不合并。

4 Experiments

4 实验

S062 · Original / 原文 · PDF p.8 · S4.p1.1

We evaluate our method on the tasks of language modeling, code generation, and image generation. For language modeling, we compare the proposed method against prior work considering the widely used generative perplexity metric. We scale the models to 1.7 billion parameters and present results on coding tasks, i.e., HumanEval *(Chen et al., 2021)*, MBPP *(Austin et al., 2021b)*, demonstrating the most promising results to date in a non-autoregressive context. In image generation, we present results for a fully discrete CIFAR10 *(Krizhevsky et al., 2009)*. Further details of the experimental setup for each model are provided in Appendix G.

我们在语言建模、代码生成和图像生成任务上评估方法。语言建模使用常见的生成困惑度指标与已有工作比较；模型扩展至 17 亿参数后，在 HumanEval（Chen 等，2021）、MBPP（Austin 等，2021b）上展示了非自回归情形当时最有潜力的编程结果。图像生成则在完全离散的 CIFAR10（Krizhevsky 等，2009）上评估。各模型的更多实验设置见附录 G。

S063 · Original / 原文 · PDF p.9 · S4.p2.1

Experimental setup. In our experiments we used the masked source, i.e., $p = \delta_m$, and trained with both unconditional coupling (U-coupling, (4)) and conditional couplings (C-coupling, (5)) with the probability path defined in equations 7, 9 and in one case 10. We trained a probability denoiser (loss in (28)) and sampled using the generating velocity in (24) and Algorithm 1. We used a particular choice of probability path scheduler κ_t, as well as corrector steps defined by a scheduler α_t and temperature annealing. We found the choice of these schedulers to be pivotal for the model's performance. In Appendix D we perform an ablation study, evaluating various scheduler choices.

实验设置。我们使用掩码源 p=δM，以无条件耦合（式 4）和条件耦合（式 5）训练，概率路径由式（7）、（9）定义，另有一个实验使用式（10）。训练概率去噪器（式 28），用式（24）的生成速度及算法 1 采样，并选择特定 κ_t 路径调度器、α_t 校正调度器及温度退火。我们发现这些调度器的选择对模型性能非常关键，附录 D 给出不同选择的消融。

4.1 Language modeling

4.1 语言建模

S064 · Original / 原文 · PDF p.9 · S4.SS1.p1.1

We experimented with our method in three settings: (i) Small model (150M parameters) - comparison to other non-autoregressive baselines in unconditional text generation; (ii) Large model (1.7B parameters) - comparison to autoregressive models in conditional text generation; and (iii) Large model (1.7B parameters) - conditional code generation. As computing exact likelihood for non-autoregressive model is a challenge, for (i),(ii) we use the generative perplexity metric (Appendix G measured with GPT2 *(Radford et al., 2019)*, Llama-2 *(Touvron et al., 2023)*, and Llama-3, and we also monitor the sentence entropy (Appendix G) to measure diversity of tokens and flag repetitive sequences, which typically yield low perplexity. Throughout our experiments we noticed entropy ≥ 6 usually corresponds to diverse texts. For (iii) we evaluated using the success rate of coding tasks.

我们考察三种设置：（i）1.5 亿参数小模型，在无条件文本生成上与其他非自回归基线比较；（ii）17 亿参数大模型，在条件文本生成上与自回归模型比较；（iii）17 亿参数大模型，进行条件代码生成。非自回归模型精确似然难以计算，故（i）、（ii）采用由 GPT2（Radford 等，2019）、Llama-2（Touvron 等，2023）、Llama-3 测量的生成困惑度（附录 G），同时监测句子熵以衡量词元多样性、识别通常困惑度较低的重复序列。实验中熵≥6 通常对应较多样的文本。对（iii）则评估编程任务成功率。

S065 · Original / 原文 · PDF p.9 · S4.SS1.p2.1

Evaluation against prior work. We evaluate our method against prior work on non-autoregressive modeling. For a fair comparison, all methods are trained on a 150M parameters models using the OpenWebText *(Gokaslan and Cohen, 2019)*

dataset. We also fix all sampling hyperparameters to the most basic settings, i.e., no temperature, top probability, corrector steps, etc. For our method we tried two paths defined by equations 9 and 10. Results are reported in Table 2, where our method outperforms all baselines in generative perplexity for all numbers of function evaluations (NFE).

与既有工作的评估。我们与非自回归建模的已有方法比较。为公平起见，全部方法都在 OpenWebText（Gokaslan 与 Cohen，2019）上训练 1.5 亿参数模型，且采样超参数均固定为最基本设置，即不使用温度、top probability 截断、校正步骤等。本文方法尝试了式（9）、（10）两条路径。表 2 表明，在所有函数评估次数（NFE）设置下，我们的生成困惑度均优于全部基线。

S066 · Original / 原文 · PDF p.9 · S4.SS1.p3.1

Conditional text generation. In this experiment, we train both C-coupling and U-coupling 1.7B parameters Discrete Flow Matching models with paths defined by (9) on a large scale data mix *(Touvron et al., 2023)*. Table 3 presents the generative perplexity of conditional generations from our method; the conditions we used are the prefixes of the first 1000 samples in OpenWeb dataset. We also compare to existing state-of-the-art autoregressive models. Our results demonstrate that our model effectively narrows the gap in generative perplexity with autoregressive models, while maintaining an entropy comparable to the recent Llama-3 8B model. Furthermore, we note the C-coupling trained model produces slightly better perplexity in conditional tasks than the U-coupling model. In Appendix I we present qualitative conditional samples produced by our U-coupling model.

条件文本生成。我们用大规模混合数据（Touvron 等，2023）及式（9）路径，分别训练 C-coupling、U-coupling 的 17 亿参数离散流匹配模型。表 3 给出条件生成的生成困惑度；条件是 OpenWeb 数据集前 1000 个样本的前缀。我们还与已有先进自回归模型比较。结果表明，模型有效缩小了与自回归模型的困惑度差距，同时保持与较新的 Llama-3 8B 相当的熵。C-coupling 训练模型在条件任务上的困惑度略好于 U-coupling。附录 I 展示 U-coupling 模型的条件生成定性样例。

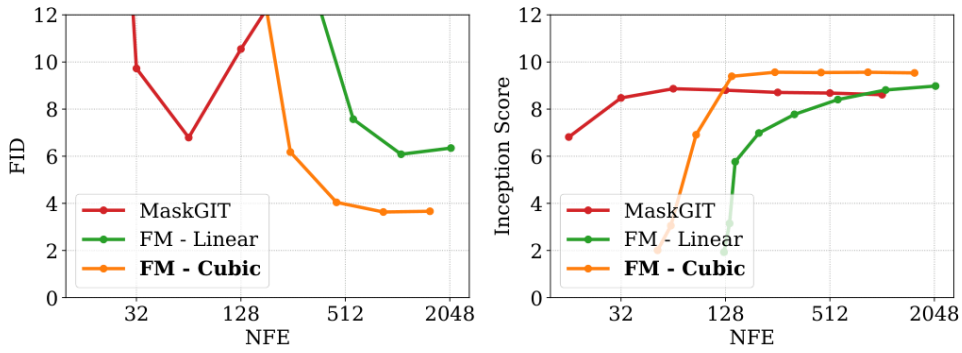
S067 · Original / 原文 · PDF p.10 · S4.SS1.p4.1

Code generation. Here we trained our basic setting of a 1.7B parameters Discrete Flow Matching model with U-coupling and path as in (9) on a code-focused data mix *(Roziere et al., 2023)*. Table 4 presents results on HumanEval and MBPP (1-shot) for pass@$\{1, 10, 25\}$. In Table 4, 'Oracle length' evaluates the performance of our model when conditioning on the length of the solution. This is done by inserting an 'end of text' token in the same position of the ground truth solution. Our method achieves non-trivial results on both tasks, which to the best of our knowledge is the first instance of a non-autoregressive method being capable of non-trivial coding tasks. In Appendix C, we analyze the proposed method for code infilling, which can be achieved as our model allows non-autoregressive generation. Lastly, in Appendix H we show qualitative examples of success and failure cases produced by our model on the coding tasks, and in Section H.3 we show examples of code infilling.

代码生成。我们用偏重代码的混合数据（Roziere 等，2023），训练基本设置的 17 亿参数离散流匹配模型，采用 U-coupling 与式（9）的路径。表 4 给出 HumanEval、MBPP（1-shot）的 Pass@{1,10,25}。"Oracle length"表示在给定解答长度的条件下评估：把文本结束词元插入到真实解答的同一位置。本文方法在两项任务都取得非平凡结果；据我们所知，这是非自回归方法首次能处理非平凡编程任务。模型支持非自回归生成，因此还可用于代码填空，分析见附录 C。附录 H 展示成功与失败案例，H.3 展示代码填空例子。

4.2 Image generation

4.2 图像生成



(a) FID. **(b)** Inception Score.

源图 / 表 / 代码：英文 PDF 第 10 页 · S4.F3

(a) FID.

（a）FID。

(b) Inception Score.

（b）Inception Score。

Figure 3: FID and Inception scores vs. number of function evaluations (NFE).

图 3：FID 与 Inception Score 随函数评估次数 NFE 的变化。

S068 · Original / 原文 · PDF p.10 · S4.SS2.p1.1

We performed a fully discrete image generation, without using any metric or neighboring information between color values. We trained an Discrete Flow Matching model with U-coupling and path as in (9) on CIFAR10 to predict discrete color value for tokens, i.e., $d = 256$, with sequence length of $N = 32 \times 32 \times 3$. For generative quality we evaluate the Fréchet Inception Distance (FID) *(Heusel et al., 2017)*. Ablations for the probability path schedulers are provided in Figure 8 in the Appendix G. In Figure 3(a) we compare our method with: (i) MaskGIT *(Chang et al., 2022)*; and (ii) *(Campbell et al., 2024)* which coincides with our method for a linear scheduler. More details in Appendix G. As can be seen in the Figure 3(a), our method outperforms both baselines, achieving 3.63 FID at 1024 NFE. In fig. 3(b) we observe a similar trend when evaluating Inception score. As discussed above, MaskGit sampling performs better for low NFE but quickly deteriorates for higher NFE. We attribute this to a bias introduced in the sampling process via the confidence mechanism.

我们进行完全离散图像生成，不使用颜色值之间的距离度量或邻域信息。在 CIFAR10 上，以 U-coupling 与式（9）路径训练离散流匹配模型，预测词元的离散颜色值，即 d=256，序列长度 N=32×32×3。生成质量用 Fréchet Inception Distance（FID；Heusel 等，2017）评估。概率路径调度器消融见附录 G 图 8。图 3(a) 比较了 MaskGIT（Chang 等，2022）和 Campbell 等（2024）；后者等同于本方法的线性调度器版本，更多细节见附录 G。本文方法优于两项基线，在 1024 NFE 时达到 FID 3.63。图 3(b) 的 Inception Score 也呈现类似趋势。正如前文所述，MaskGIT 在低 NFE 时更好，但随 NFE 增大迅速恶化；我们将此归因于置信度机制引入的采样偏差。

5 Conclusions and future work

5 结论与未来工作

S069 · Original / 原文 · PDF p.10 · S5.p1.1

We introduce Discrete Flow Matching, a generalization of continuous flow matching and discrete flows that provides a large design space of discrete non-autoregressive generative models. Searching within this space we were able to train large scale

language models that produce generated text with an improved generative perplexity compared to current non-autoregressive methods and able to solve coding tasks at rates not achievable before with non-autoregressive models, as far as we are aware. While reducing the number of network evaluations required to generate a discrete sample compared to autoregressive models, Discrete Flow Matching still does not achieve the level of sampling efficiency achieved by its continuous counterpart, flagging an interesting future work direction. Another interesting direction is to explore the space of probability paths in (8) (or a generalization of which) beyond what we have done in this paper. We believe discrete non-autoregressive models have the potential to close the gap and even surpass autoregressive models as well as unlock novel applications and use cases. As our work introduces an alternative modeling paradigm to discrete sequential data such as language and code, we feel it does not introduce significant societal risks beyond those that already exist with previous large language models.

我们提出离散流匹配，它推广了连续流匹配与离散流，为离散非自回归生成提供广阔设计空间。在其中探索后，我们训练出大规模语言模型，其文本生成困惑度优于当时非自回归方法，并以据我们所知此前非自回归模型达不到的成功率完成编程任务。虽然相对自回归模型减少了生成一个离散样本所需的网络评估次数，其采样效率仍未达到连续流匹配的水平，这是值得研究的方向。另一个方向是进一步探索式（8）及其推广所允许的概率路径空间。我们认为离散非自回归模型可能缩小乃至超越与自回归模型的差距，并带来新应用。本文针对语言、代码等离散序列提供另一建模范式，作者认为它没有引入超出现有大语言模型的显著额外社会风险。

References
参考文献

- Ahn et al. (2024) Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.
- Albergo and Vanden-Eijnden (2022) Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.
- Austin et al. (2021a) 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, 2021a.
- Austin et al. (2021b) Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.
- Besnier and Chen (2023) Victor Besnier and Mickael Chen. A pytorch reproduction of masked generative image transformer, 2023.
- Blattmann et al. (2023) Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.
- Brown et al. (2020) Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Campbell et al. (2022) Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.
- Campbell et al. (2024) 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. *arXiv preprint arXiv:2402.04997*, 2024.
- Chang et al. (2022) Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.
- Chang et al. (2023) Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

- Chen et al. (2021) Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Chen et al. (2022) Ting Chen, Ruixiang Zhang, and Geoffrey E. Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *ArXiv*, 2022.
- Copet et al. (2024) Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36, 2024.
- Dhariwal and Nichol (2021) Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- Dieleman et al. (2022) Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.
- Esser et al. (2024) Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.
- Ferruz and Höcker (2022) Noelia Ferruz and Birte Höcker. Controllable protein design with language models. *Nature Machine Intelligence*, 4(6):521–532, 2022.
- Ghazvininejad et al. (2019) Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.
- Gokaslan and Cohen (2019) Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.
- Han et al. (2022) Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.
- Hassid et al. (2024) Michael Hassid, Tal Remez, Tu Anh Nguyen, Itai Gat, Alexis Conneau, Felix Kreuk, Jade Copet, Alexandre Defossez, Gabriel Synnaeve, Emmanuel Dupoux, et al. Textually pretrained speech language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- He et al. (2022) Zhengfu He, Tianxiang Sun, Kuan Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *Annual Meeting of the Association for Computational Linguistics*, 2022.
- Heusel et al. (2017) Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- Ho et al. (2020) Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- Hoogeboom et al. (2021) 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.
- Imani et al. (2023) Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, 2023.
- Kreuk et al. (2022) Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.
- Krizhevsky et al. (2009) Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *arXiv*, 2009.
- Li et al. (2023) Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

- Li et al. (2022) Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-lm improves controllable text generation. *ArXiv*, 2022.
- Lin et al. (2022) Zheng-Wen Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Weizhu Chen, and Nan Duan. Genie : Large scale pre-training for generation with diffusion model. In *ArXiv*, 2022.
- Lipman et al. (2022) Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- Liu et al. (2022) Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.
- Lou et al. (2023) Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.
- Lovelace et al. (2022) Justin Lovelace, Varsha Kishore, Chao gang Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. *ArXiv*, 2022.
- Madani et al. (2023) Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023.
- Norris (1998) James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.
- Peebles and Xie (2022) William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.
- Pooladian et al. (2023) Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.
- Radford et al. (2019) Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. https://api.semanticscholar.org/CorpusID:160025533.
- Rombach et al. (2022) 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.
- Romera-Paredes et al. (2024) Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.
- Roziere et al. (2023) Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.
- Savinov et al. (2021) Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. *arXiv preprint arXiv:2112.06749*, 2021.
- Singer et al. (2022) Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.
- Song and Ermon (2019) Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- Song et al. (2020) Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- Stark et al. (2024) Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.
- Su et al. (2024) Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

- Tong et al. (2023) Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.
- Touvron et al. (2023) Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Zhang et al. (2024) Qiang Zhang, Keyang Ding, Tianwen Lyv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, et al. Scientific large language models: A survey on biological & chemical domains. *arXiv preprint arXiv:2401.14656*, 2024.
- Zhao et al. (2023) Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- Zheng et al. (2024) Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.
- Ziv et al. (2024) Alon Ziv, Itai Gat, Gael Le Lan, Tal Remez, Felix Kreuk, Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. Masked audio generation using a single non-autoregressive transformer. *arXiv preprint arXiv:2401.04577*, 2024.

Appendix A Related works, continuation

附录 A 其他相关工作

S070 · Original / 原文 · PDF p.14 · A1.p1.1

We provide here some more details on relevant related works.

这里进一步介绍相关研究。

Continuous diffusion and flows.

连续扩散与流

S071 · Original / 原文 · PDF p.14 · A1.SS0.SSS0.Px1.p1.1

Another line of works has been exploring the use of continuous space diffusion for discrete data, typically operating in the logits space *(Dieleman et al., 2022; Li et al., 2022; Han et al., 2022; Lin et al., 2022; Chen et al., 2022)*. An additional body of work has been focusing on the adoption of latent diffusion-like modeling *(Lovelace et al., 2022; He et al., 2022)*. *Stark et al. (2024)* proposed to learn a continuous Flow Matching on the probability simplex with Dirichlet paths.

另一条研究路线是在连续空间对离散数据进行扩散，通常操作 logits 空间（Dieleman 等，2022；Li 等，2022；Han 等，2022；Lin 等，2022；Chen 等，2022）。另一些工作采用类似潜变量扩散的建模（Lovelace 等，2022；He 等，2022）。Stark 等（2024）提出利用 Dirichlet 路径，在概率单纯形上学习连续流匹配。

Autoregressive modeling.

自回归建模

S072 · Original / 原文 · PDF p.14 · A1.SS0.SSS0.Px2.p1.1

Autoregressive models have been a significant area of focus in recent years, particularly in the context of natural language processing and machine learning *(Zhao et al., 2023)*. Autoregressive modeling, in its most fundamental form, utilizes the chain rule to learn the joint sequence probability by breaking it down into next-token conditional probabilities. GPT-2 *(Radford et al., 2019)*, showcased the power of autoregressive language models in generating coherent and contextually relevant text over long passages. Its successor, GPT-3 *(Brown et al., 2020)*, further pushed the boundaries, demonstrating impressive performance across a range of tasks without task-specific training data. Later models were adapted to other domains such as, code *(Roziere et al., 2023; Li et al., 2023; Chen et al., 2021)*, biology *(Zhang et al., 2024; Ferruz and Höcker, 2022; Madani et al., 2023)*, math *(Romera-Paredes et al., 2024; Imani et al., 2023; Ahn et al., 2024)*, audio *(Kreuk et al., 2022; Copet et al., 2024; Hassid et al., 2024)* and more.

近年来，自回归模型一直是自然语言处理与机器学习的重要方向（Zhao 等，2023）。其基本形式利用概率链式法则，把序列联合概率拆成逐个下一词元的条件概率来学习。GPT-2（Radford 等，2019）展示了生成连贯且符合上下文长文本的能力；GPT-3（Brown 等，2020）进一步表明，即使没有特定任务训练数据，也可在多种任务上表现良好。后续模型被扩展到代码（Roziere 等，2023；Li 等，2023；Chen 等，2021）、生物学（Zhang 等，2024；Ferruz 与 Höcker，2022；Madani 等，2023）、数学（Romera-Paredes 等，2024；Imani 等，2023；Ahn 等，2024）、音频（Kreuk 等，2022；Copet 等，2024；Hassid 等，2024）等领域。

Masked generative modeling.

掩码生成建模

S073 · Original / 原文 · PDF p.14 · A1.SS0.SSS0.Px3.p1.1

Masked generative modeling proposes to mask a variable portion of the input sequence and training a model to predict this masked section. *Ghazvininejad et al. (2019)* proposed Mask-Predict, a masked language modeling with parallel decoding. *Savinov et al. (2021)* extended the mask-modeling approach by employing an additional loss term that incorporates rolling model predictions. MaskGIT *(Chang et al., 2022)* followed a similar path, for the task of class-conditioned image synthesis, *Chang et al. (2023)* extended this approach to high-quality textually guided image generation over low-resolution images followed by a super-resolution module. Recently, *Ziv et al. (2024)* proposed a text-to-music method, which relies on the MaskGIT foundations while observing that span masking boosts the quality of the generated sequence significantly.

掩码生成建模对输入序列中可变比例的部分做掩码，训练模型预测被遮挡部分。Ghazvininejad 等（2019）的 Mask-Predict 是并行解码的掩码语言模型；Savinov 等（2021）通过纳入滚动模型预测的额外损失项扩展此路线。MaskGIT（Chang 等，2022）沿类似思路进行类别条件图像合成，Chang 等（2023）又将其扩展为先生成低分辨率图像、再超分辨率的高质量文本引导图像生成。近期 Ziv 等（2024）基于 MaskGIT 提出文本到音乐方法，并观察到连续片段掩码显著提高生成序列质量。

Appendix B Further implementation details
附录 B 更多实现细节

Safe sampling.

安全采样

S074 · Original / 原文 · PDF p.14 · A2.SS0.SSS0.Px1.p1.1

When sampling according to Algorithm 1 using the generating probability velocity in (22), an arbitrary step size $h > 0$ can make some probabilities in $\delta_{X_t^i}(\cdot) + hu_t^i(\cdot, X_t)$ negative and consequently require clamping and injecting further error into the sampling process that can in turn accumulate to a non-negligible global sampling error. A simple fix that guarantees a valid probability distribution while keeping the $o(h)$ sampling error at the relatively manageable price of potentially more function evaluations is using the following adaptive step size in Algorithm 1: at time $t \in [0, 1)$ use

按算法 1、式（22）的概率速度采样时，任意步长 h>0 可能使 $\delta_{X_t^i}(\cdot) + hu_t^i(\cdot, X_t)$ 的某些概率变负，从而不得不裁剪，额外误差可能累积成不可忽略的全局采样误差。简单修复是使用下列自适应步长：它保证概率分布有效，保留 o(h) 采样误差，代价是可能增加函数评估次数。在 $t \in [0, 1)$ 时取下式。

$$h_{\text{adaptive}} = \min\left\{h, \min_i \left|\frac{\kappa_t^{i,\ell}}{\dot{\kappa}_t^{i,\ell}}\right|\right\}. \tag{29}$$

S075 · Original / 原文 · PDF p.14 · A2.SS0.SSS0.Px1.p1.2

As can be verified with the general probability velocity formula in (22), the above choice for h_{adaptive} guarantees $\delta_{X_t^i}(\cdot) + hu_t^i(\cdot, X_t)$ is a valid PMF. As mostly used in this paper, for the probability denoiser parameterization ((24)) the adaptive step is

利用一般概率速度公式（22）可验证，此 h_adaptive 保证更新后的分布为有效 PMF。对于本文主要采用的概率去噪器参数化（式 24），自适应步长如下。

$$h_{\text{adaptive}} = \min\left\{h, \frac{1-\kappa_t}{\dot{\kappa}_t}\right\}. \tag{30}$$

S076 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px1.p1.3

With the corrector sampling (equations 26 and 51) we have the adaptive step:

若采用校正采样（式 26、51），则自适应步长如下。

$$h_{\text{adaptive}} = \min\left\{h, \left[\frac{\alpha_t \dot{\kappa}_t}{1-\kappa_t} + \frac{\beta_t \dot{\kappa}_t}{\kappa_t}\right]^{-1}\right\}. \tag{31}$$

Conditioning.

条件化

S077 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px2.p1.1

In our unconditional coupling (U-coupling), see (5), we define the conditioning pattern based on prefixes of random length $N_0 < N$, i.e.,

在我们的无条件耦合（U-coupling；原文此处引用式 5）中，条件模式按随机长度 $N_0<N$ 的前缀定义如下。

$$\mathbb{I} = (\overbrace{1,\dots,1}^{N_0}, \overbrace{0,\dots,0}^{N-N_0}).$$

S078 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px2.p1.2

During the training phase, we sample $N_0 \sim \mathcal{U}(0, N)$ and adjust the input sequence in accordance with the mask $\mathbb{I}$.

训练阶段，采样 $N_0 \sim \mathcal{U}(0, N)$，按掩码 I 相应调整输入序列。

S079 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px2.p2.1

During conditional sampling with Algorithm 1 we replace, after each update step, the relevant tokens with the conditioned ones, i.e., $\tilde{X} = \mathbb{I} \odot Y + (\mathbf{1} - \mathbb{I}) \odot X$, where X is the current sample, Y is the condition, and $\mathbb{I}$ is the condition's mask.

条件采样时，每次算法 1 更新之后，把相关词元换回给定条件，即 $\tilde{X} = I \odot Y + (\mathbf{1} - I) \odot X$；X 是当前样本，Y 是条件，I 是条件掩码。

NFE bound.

NFE 上界

S080 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px3.p1.1

For mask modeling, i.e., $p = \delta_m$, we have seen that the probability denoiser is time-independent (see Proposition 6). Consequently, when sampling with Algorithm 1 and u_t from (24) without corrector sampling one is not required to recompute the forward pass $p_{1|t}(\cdot|X_t)$ if X_t is identical to X_{t-h} (i.e., no m has been unmasked). This means that the NFE of Algorithm 1 in this case is bounded by the number of tokens N.

对于掩码建模 p=δM，命题 6 已说明概率去噪器与时间无关。因此，用式（24）且不采用校正采样时，如果 X_t 与 X_{t-h} 相同，即没有 mask 被解开，则不必重新计算 $p_{1|t}(\cdot|X_t)$ 的前向传播。此时算法 1 的 NFE 上界就是词元数 N。

Post training scheduler change.

训练后改变调度器

S081 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px4.p1.1

For a trained posterior $\hat{w}_t(x^i|z)$ of a conditional probability path as in (9) with a scheduler κ_t, the velocity is given by equations 24 or 25, where $\hat{w}_t(x^i|z)$ is either $p_{1|t}(x^i|z)$ or $p_{0|t}(x^i|z)$ respectively. In this case, we can apply the velocities in equations 24 and 25 for sampling with any scheduler κ'_t, using the change of scheduler formula for posteriors,

若条件路径为式（9）、调度器为 κ_t，其训练好的后验为 $\hat{w}_t(x^i|z)$，则式（24）、（25）分别给出速度，后验相应是 $p_{1|t}$ 或 $p_{0|t}$。这时可通过下列后验调度器变换公式，使用任意新调度器 κ'_t 按原速度公式采样。

$$\hat{w}'_t(x^i|z) = \hat{w}_{t'}(x^i|z), \tag{32}$$

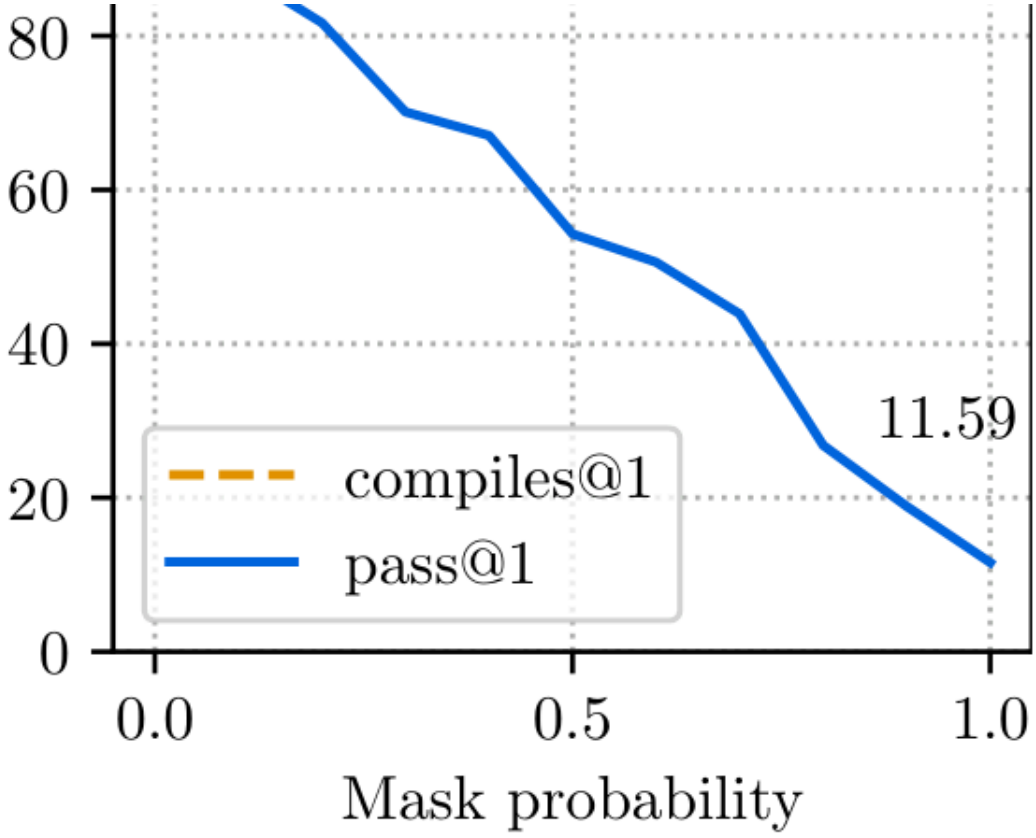
S082 · Original / 原文 · PDF p.15 · A2.SS0.SSS0.Px4.p1.2

where $\hat{w}'_t(x^i|z)$, is the posterior of the scheduler κ'_t, $t' = \kappa^{-1}_{\kappa'_t}$, and κ^{-1} is the inverse of κ. The scheduler change formula in (32) is proved in Proposition 8. We note that by Proposition 6, for mask modeling, i.e., $p = \delta_m$, the posterior $\hat{w}_t(x^i|z)$ is time independent. Hence, in that case, the posterior is not affected by a scheduler change.

这里 $\hat{w}'_t(x^i|z)$ 是 κ'_t 对应的后验，$t' = \kappa^{-1}(\kappa'_t)$，$\kappa^{-1}$ 为 κ 的逆函数。式（32）的调度器变换在命题 8 中证明。根据命题 6，掩码源 $p=\delta_M$ 的后验与时间无关，所以这种情形下，改变调度器不会改变后验。

Appendix C Code infilling

附录 C 代码填空



源图 / 表 / 代码：英文 PDF 第 15 页 · A3.F4

Figure 4: Pass@1 and compiles@1 scores for the 1.5B parameter models as a function of the input masking rations on HumanEval.

图 4：HumanEval 上，原文所称 15 亿参数模型的 Pass@1、compiles@1 随输入掩码比例的变化。

S083 · Original / 原文 · PDF p.15 · A3.p1.1

We additionally evaluate the proposed method considering the task of code infilling. In which, we are provided with an input prompt that contains various spans of masked tokens, and our goal is to predict them based on the unmasked ones. See Figure 1 (middle and right sub-figures) for a visual example. Notice, this evaluation setup is the most similar to the training process.

我们还评估代码填空：输入提示包含若干被掩码的片段，目标是依据未被掩码的部分预测这些片段，图 1 中、右子图提供直观例子。注意，这种评估设置最接近训练过程。

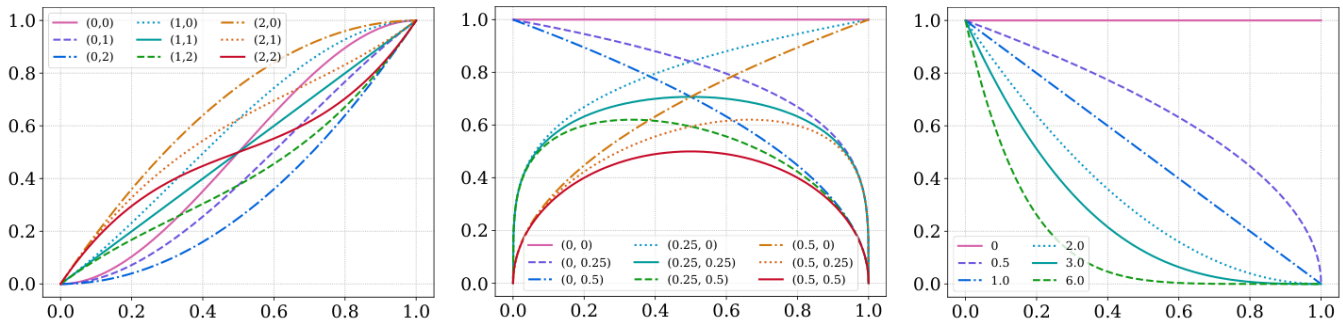
S084 · Original / 原文 · PDF p.16 · A3.p2.1

For that, we randomly mask tokens with respect to several masking rations, $p \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$, from HumanEval and report both pass@1 and compiles@1 metrics. For the purpose of this analysis, we provide the oracle length for each masked span. In other words, the model predicts the masked tokens for already given maks length. Results for the 1.5B parameters models can be seen in Figure 4. As expected, both pass@1 and compiles@1 keep improving as we decrease the level of input masking. Interestingly, when considering the fully masked sequence, providing the oracle prediction length significantly improves the pass@1 scores (6.7 vs. 11.6).

我们按掩码比例 $p \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$ 随机掩码 HumanEval 的词元，报告 Pass@1 与 compiles@1。此分析给每个掩码片段提供 oracle 长度，即模型只须预测已经给定长度的缺失词元。图 4 给出原文所称 15 亿参数模型的结果。符合预期，输入掩码比例降低时，两项指标都改善。特别是对全掩码序列，提供 oracle 预测长度显著改善了 Pass@1：6.7 对 11.6。

Appendix D Ablations

附录 D 消融实验



(a) Path scheduler, cubic poly. **(b)** Corrector scheduler $t^a(1-t)^b$. **(c)** Temperature scheduler $(1-t)^a$.

源图 / 表 / 代码：英文 PDF 第 16 页 · A4.fig1

(a) Path scheduler, cubic poly.

（a）路径调度器：三次多项式。

(b) Corrector scheduler $t^a(1-t)^b$.

（b）校正调度器 $t^a(1-t)^b$。

(c) Temperature scheduler $(1-t)^a$.

（c）温度调度器 $(1-t)^a$。

Train and sampling path scheduler choice (κ_t).

训练与采样路径调度器选择（κ_t）

S085 · Original / 原文 · PDF p.16 · A4.SS0.SSS0.Px1.p1.1

We study how the choice of the probability path scheduler affects the model performance. For that, we consider a parametric family of cubic polynomial with parameters a, b:

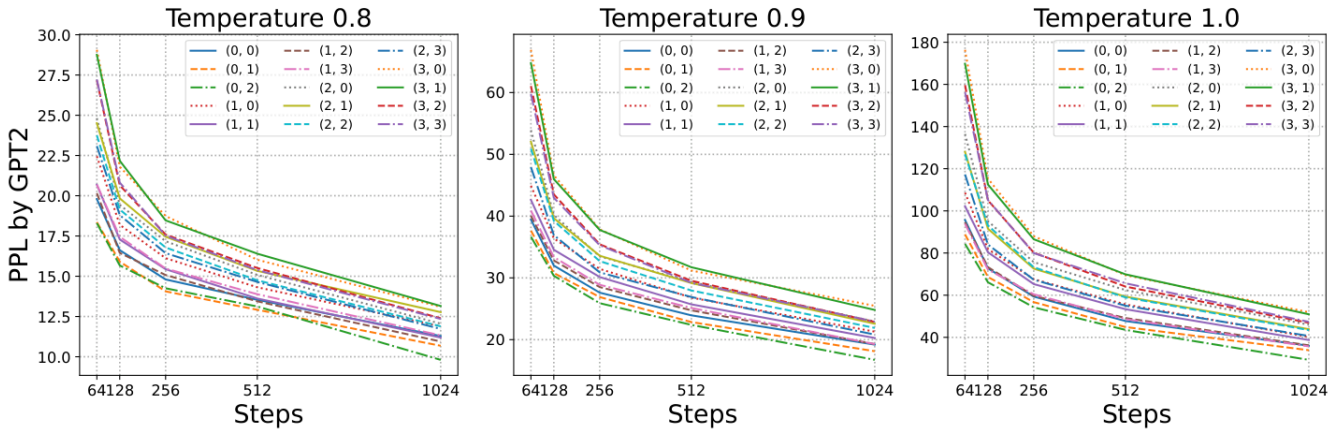
我们研究概率路径调度器的选择如何影响性能，考察由参数 a、b 控制的三次多项式族如下。

$$\kappa_t \triangleq -2t^3 + 3t^2 + a(t^3 - 2t^2 + t) + b(t^3 - t^2). \tag{33}$$

S086 · Original / 原文 · PDF p.16 · A4.SS0.SSS0.Px1.p1.2

Note that $\kappa_0 = 0$ and $\kappa_1 = 0$ and a and b are setting the derivative of κ_t at $t = 0$ and $t = 1$, respectively. We visualize this κ_t with choices of $a, b \in \{0, 1, 2\}$ in Figure 5(a).

原文此处写 $\kappa_0=0$、$\kappa_1=0$；a、b 分别设置 κ_t 在 t=0、t=1 的导数。图 5(a) 可视化了 $a, b \in \{0, 1, 2\}$ 的调度器。译注：由上方公式实际得到 $\kappa_1=1$，此处是原文端点笔误。



源图 / 表 / 代码：英文 PDF 第 16 页 · A4.F6

Figure 6: Path scheduler choice during training using various of constant temperature values.

图 6：不同固定温度下，训练时路径调度器选择的效果。

S087 · Original / 原文 · PDF p.16 · A4.SS0.SSS0.Px1.p2.1

To test the effect of path schedulers in training we have trained 150M parameters models for all choices of $a, b \in \{0, 1, 2, 3\}$. We then generate 1000 samples from each model. The samples are computed using Algorithm 1 with the path scheduler the model was trained on, and with temperature levels $\tau \in \{0.8, 0.9, 1\}$, where temperature is applied via

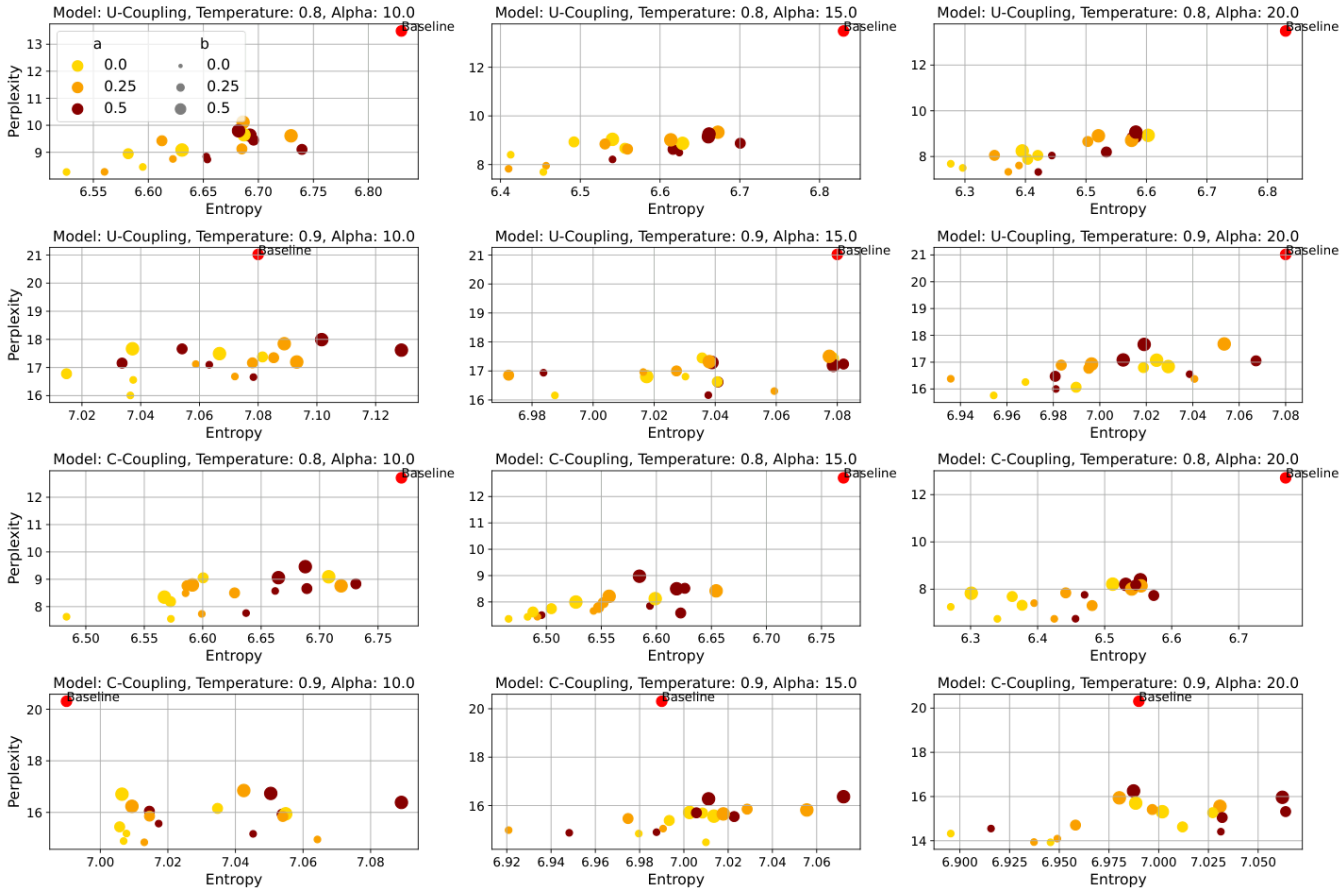
为检验训练路径调度器的影响，我们对全部 $a, b \in \{0, 1, 2, 3\}$ 组合训练 1.5 亿参数模型，每个模型生成 1000 个样本。采样使用算法 1、与训练相同的调度器，以及 $\tau \in \{0.8, 0.9, 1\}$ 的温度；温度按下式作用于概率。

$$p^{\tau}_{1|t}(x^i|X_t) = \tau^{-1} \log p_{1|t}(x^i|X_t). \tag{34}$$

S088 · Original / 原文 · PDF p.16 · A4.SS0.SSS0.Px1.p2.2

We then evaluate the generative perplexity of these samples with GPT-2. Figure 6 shows the results. The graphs indicate that, in the context of text modality, the cubic polynomial scheduler with $a \equiv 0, b \equiv 2$ (equivalent to a square function) achieves the highest performance. Consequently, we exclusively used this scheduler for the language models.

随后用 GPT-2 评估这些样本的生成困惑度，结果见图 6。文本模态下，a=0、b=2 的三次多项式调度器等价于平方函数，表现最佳。因此语言模型统一采用该调度器。



源图 / 表 / 代码：英文 PDF 第 17 页 · A4.F7

Figure 7: Corrector scheduler ablation.

图 7：校正调度器消融。

Corrector scheduler.

校正调度器

S089 · Original / 原文 · PDF p.16 · A4.SS0.SSS0.Px2.p1.1

In our experiments we only applied corrector sampling to our large models (U-coupling and C-coupling; 1.7B parameters). We used the optimal path schedulers from previous section and considered the following parametric family of schedulers for the corrector sampling:

实验只对大模型（U-coupling、C-coupling，17 亿参数）采用校正采样。使用上一节最优路径调度器，并考察下列参数化的校正调度器族。

$$\alpha_t = 1 + \alpha t^a (1-t)^b, \tag{35}$$

S090 · Original / 原文 · PDF p.17 · A4.SS0.SSS0.Px2.p1.2

where, we set $\beta_t = \alpha_t - 1$ and generate 1000 samples using Algorithm 1 with parameter values $a, b \in \{0, 0.25, 0.5\}$ and $\alpha \in \{10, 15, 20\}$. We then evaluated generative perplexity for these samples with Llama-2, showing results in Figure 7. These plots indicate that smaller values of a and b result in lower perplexity values, albeit with somewhat reduced entropy. We therefore opted for setting $a = b = 0.25$ that strikes a good balance between perplexity and entropy.

其中设 $\beta_t=\alpha_t-1$，采用 $a, b \in \{0, 0.25, 0.5\}$、$\alpha \in \{10, 15, 20\}$，以算法 1 生成 1000 个样本，再用 Llama-2 评估困惑度，结果见图 7。较小 a、b 对应较低困惑度，但熵也略降。因此选取 a=b=0.25，在困惑度与熵之间取得平衡。

Temperature scheduling.

温度调度

S091 · Original / 原文 · PDF p.17 · A4.SS0.SSS0.Px3.p1.1

For temperature sampling, we consider the following scheduler:

温度采样采用下列调度器。

$$\tau_t = \tau(1-t)^2. \tag{36}$$

Appendix E Theory and proofs

附录 E 理论与证明

E.1 Computation of the discrete divergence

E.1 离散散度的计算

S092 · Original / 原文 · PDF p.17 · A5.SS1.p1.1

We present the computation of the discrete divergence in (18), i.e.,

这里计算式（18）的离散散度，即下式。

$$\mathrm{div}_x(p_t u_t) = -\sum_{z\in\mathcal{D}} p_t(z)\left[\sum_{i=1}^{N}\delta_z(x^{\bar{i}})u_t^i(x^i,z)\right]. \tag{37}$$

S093 · Original / 原文 · PDF p.17 · A5.SS1.p2.1

Computing the discrete divergence ((18)) of the flux $p_t u_t$ at a state x amounts to adding outgoing flux from x and subtracting the incoming flux into x. Using the fact that $\delta_z(x^{\bar{i}}) = 1$ if and only if $z = x$ or z differs from x only at the i-th token, gives:

计算状态 x 处通量 $p_t u_t$ 的离散散度（式 18），就是加上离开 x 的通量、减去进入 x 的通量。利用 $\delta_z(x^{\bar{i}}) = 1$ 当且仅当 z=x 或 z 与 x 仅在第 i 个词元不同，得到下列计算。

$$\begin{aligned}
\mathrm{div}_x(p_t u_t) &= \sum_{z\in\mathcal{D}}\sum_{i=1}^{N}\delta_x(z^{\bar{i}})\left(p_t(x)u_t^i(z^i,x) - p_t(z)u_t^i(x^i,z)\right)\\
&= p_t(x)\sum_{i=1}^{N}\sum_{z^i}\overbrace{\left[\sum_{z^{\bar{i}}}\delta_x(z^{\bar{i}})\right]}^{=1}u_t^i(z^i,x) - \sum_{z\in\mathcal{D}}\sum_{i=1}^{N}\delta_x(z^{\bar{i}})p_t(z)u_t^i(x^i,z)\\
&= p_t(x)\sum_{i=1}^{N}\overbrace{\left[\sum_{z^i}u_t^i(z^i,x)\right]}^{=0} - \sum_{z\in\mathcal{D}}\sum_{i=1}^{N}\delta_x(z^{\bar{i}})p_t(z)u_t^i(x^i,z) \quad \text{[uses (13)]}\\
&= -\sum_{z\in\mathcal{D}}\sum_{i=1}^{N}\delta_x(z^{\bar{i}})p_t(z)u_t^i(x^i,z),
\end{aligned}$$

S094 · Original / 原文 · PDF p.18 · A5.SS1.p2.2

that gives (37) after noting that $\delta_x(z^{\bar{i}}) = \delta_z(x^{\bar{i}})$.

再注意 $\delta_x(z^{\bar{i}}) = \delta_z(x^{\bar{i}})$，即得到式（37）。

E.2 Conditional velocities lead to marginal velocities

E.2 从条件速度到边缘速度

S095 · Original / 原文 · PDF p.18 · A5.SS2.p1.1

We provide a simple proof for Theorem 2, originally proved in *Campbell et al. (2024)*:

Theorem 2.
Given a conditional probability velocity $u_t^i(x^i, X_t|x_0, x_1)$ generating a conditional probability path $p_t(x|x_0, x_1)$, the marginal velocity defined by

这里给出定理 2 的简短证明，该结果最早由 Campbell 等（2024）证明。重述定理：给定生成条件路径 $p_t(x|x_0, x_1)$ 的条件概率速度 $u_t^i(x^i, X_t|x_0, x_1)$，下式定义的边缘速度：

$$u_t^i(x^i, X_t) = \sum_{x_0, x_1 \in \mathcal{D}} u_t^i(x^i, X_t|x_0, x_1) p_t(x_0, x_1|X_t), \tag{38}$$

S096 · Original / 原文 · PDF p.18 · A5.SS2.p1.2

generates the marginal probability path $p_t(x)$, where by Bayes' rule

生成边缘概率路径 $p_t(x)$，其中由 Bayes 公式可得下式。

$$p_t(x_0, x_1|X_t) = \frac{p_t(X_t|x_0, x_1)\pi(x_0, x_1)}{p_t(x)}. \tag{39}$$

Proof (Theorem 2). 证明（定理 2）

S097 · Original / 原文 · PDF p.18 · A5.Thmtheorem1.p1.1

We start by taking the time derivative of the marginal probability path, $p_t(x) = \sum_{x_0, x_1} p_t(x^i|x_0, x_1)\pi(x_0, x_1)$, as follows,

从边缘概率路径对时间求导开始，计算如下。原文此处写 $p_t(x) = \sum_{x_0, x_1} p_t(x^i|x_0, x_1)\pi(x_0, x_1)$，其展开应按式（7）理解为联合条件路径。

$$\begin{aligned}
\dot{p}_t(x) &= \sum_{x_0, x_1} \dot{p}_t(x|x_0, x_1)\pi(x_0, x_1) \\
&= \sum_{x_0, x_1} \left(\sum_z p_t(z|x_0, x_1) \left[\sum_{i=1}^N \delta_z(x^{\bar{i}}) u_t^i(x^i, z|x_0, x_1) \right] \right) \pi(x_0, x_1) \quad \text{[uses (16)]} \\
&= \sum_z p_t(z) \left[\sum_{i=1}^N \delta_z(x^{\bar{i}}) \left(\sum_{x_0, x_1} u_t^i(x^i, z|x_0, x_1) \frac{p_t(z|x_0, x_1)\pi(x_0, x_1)}{p_t(z)} \right) \right] \\
&= \sum_z p_t(z) \left[\sum_{i=1}^N \delta_z(x^{\bar{i}}) u_t^i(x^i, z) \right] \\
&= -\mathrm{div}_x(p_t u_t)
\end{aligned}$$

S098 · Original / 原文 · PDF p.18 · A5.Thmtheorem1.p1.2

Now since $u_t^i(x^i, z)$ is a convex combinations of $u_t^i(x^i, z|x_0, x_1)$ and these satisfy (13) then also $u_t^i(x^i, X_t)$ satisfies (13).

边缘速度 $u_t^i(x^i, z)$ 是各条件速度 $u_t^i(x^i, z|x_0, x_1)$ 的凸组合，而后者满足式（13），故 $u_t^i(x^i, X_t)$ 也满足式（13）。

E.3 Probability velocities generating conditional probability paths

E.3 条件路径的生成概率速度

S099 · Original / 原文 · PDF p.19 · A5.SS3.p1.1

Equation 22 with the coefficients $a_t^{i,j}$ and b_t^i are provided below,

下面给出式（22）及其系数 $a_t^{i,j}$、b_t^i。

$$u_t^i(x^i, X_t|x_0, x_1) = \sum_{j=1}^{m} \overbrace{\left[\dot\kappa_t^{i,j} - \kappa_t^{i,j}\frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}}\right]}^{a_t^{i,j}} w^j(x^i|x_0, x_1) + \overbrace{\left[\frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}}\right]}^{b_t^i}\delta_{X_t}(x^i), \tag{40}$$

S100 · Original / 原文 · PDF p.19 · A5.SS3.p1.2

where

其中系数如下。

$$\ell = \ell(i,t) \stackrel{\text{def}}{=} \arg\min_{j\in[m]} \left[\dot\kappa_t^{i,j}/\kappa_t^{i,j}\right]. \tag{41}$$

Theorem 3.

S101 · Original / 原文 · PDF p.19 · A5.SS3.p1.4

[Probability velocity of conditional paths] A generating probability velocity for the conditional paths $p_t(x|x_0, x_1)$ defined in equations 7 and 8 is

重述定理 3（条件路径的概率速度）：生成式（7）、（8）所定义条件概率路径 $p_t(x|x_0, x_1)$ 的一个概率速度如下。

$$u_t^i(x^i, X_t|x_0, x_1) = \sum_{j=1}^{m} a_t^{i,j} w^j(x^i|x_0, x_1) + b_t^i \delta_{X_t}(x^i), \tag{42}$$

S102 · Original / 原文 · PDF p.19 · A5.SS3.p1.5

with $a_t^{i,j} = \dot\kappa_t^{i,j} - \kappa_t^{i,j}\dot\kappa_t^{i,\ell}/\kappa_t^{i,\ell}$, and $b_t^i = \dot\kappa_t^{i,\ell}/\kappa_t^{i,\ell}$ where $\ell = \arg\min_{j\in[m]}\left[\dot\kappa_t^{i,j}/\kappa_t^{i,j}\right]$.

其中 $a_t^{i,j} = \dot\kappa_t^{i,j} - \kappa_t^{i,j}\dot\kappa_t^{i,\ell}/\kappa_t^{i,\ell}$，$b_t^i = \dot\kappa_t^{i,\ell}/\kappa_t^{i,\ell}$，且 $\ell = \arg\min_{j\in[m]}(\dot\kappa_t^{i,j}/\kappa_t^{i,j})$。

Proof (Theorem 3). 证明（定理 3）

S103 · Original / 原文 · PDF p.19 · A5.Thmtheorem2.p1.1

First, let us show that (40) satisfies the conditions in (13): Fix $X_t \in \mathcal{D}$, and

先证明式（40）满足式（13）的条件。固定任意 $X_t \in \mathcal{D}$，有下式。

$$\begin{aligned}
\sum_{x^i} u_t^i(x^i, X_t|x_0, x_1) &= \sum_{x^i}\left[\sum_{j=1}^{m}\left[\dot\kappa_t^{i,j} - \kappa_t^{i,j}\frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}}\right] w^j(x^i|x_0, x_1) + \frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}}\delta_{X_t}(x^i)\right] \\
&= \sum_{j=1}^{m}\left[\dot\kappa_t^{i,j} - \kappa_t^{i,j}\frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}}\right] + \frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}} \\
&= \sum_{j=1}^{m}\dot\kappa_t^{i,j} + \frac{\dot\kappa_t^{i,\ell}}{\kappa_t^{i,\ell}}\left(1 - \sum_{j=1}^{m}\kappa_t^{i,j}\right) \quad \text{[uses (44)]} \\
&= 0.
\end{aligned}$$

S104 · Original / 原文 · PDF p.19 · A5.Thmtheorem2.p1.2

and for $x^i \neq X_t^i$ we have

对 $x^i \neq X_t^i$，有下式。

$$u_t^i(x^i, X_t|x_0, x_1) = \sum_{j=1}^{m} \left[\frac{\dot{\kappa}_t^{i,j}}{\kappa_t^{i,j}} - \frac{\dot{\kappa}_t^{i,\ell}}{\kappa_t^{i,\ell}} \right] \kappa_t^{i,j} w^j(x^i|x_0, x_1) \geq 0 \tag{43}$$

S105 · Original / 原文 · PDF p.19 · A5.Thmtheorem2.p1.3

since $\kappa_t^{i,j} \geq 0$, $\hat{w}_t(x^i|z) \geq 0$, and $\frac{\dot{\kappa}_t^{i,j}}{\kappa_t^{i,j}} - \frac{\dot{\kappa}_t^{i,\ell}}{\kappa_t^{i,\ell}} \geq 0$ since $\ell = \arg\min_{j\in[m]} \frac{\dot{\kappa}_t^{i,j}}{\kappa_t^{i,j}}$. Second, we show that u_t satisfies the Continuity Equation ((16)). To that end we write (8) as

因为 $\kappa_t^{i,j} \geq 0$、$\hat{w}_t(x^i|z) \geq 0$，且由 ℓ 的最小值定义，$\dot{\kappa}_t^{i,j}/\kappa_t^{i,j} - \dot{\kappa}_t^{i,\ell}/\kappa_t^{i,\ell} \geq 0$。其次证明 u_t 满足连续性方程（16），为此把式（8）改写如下。

$$w^\ell(x^i|x_0, x_1) = \frac{1}{\kappa_t^{i,\ell}} \left[p_t(x^i|x_0, x_1) - \sum_{j\neq\ell} \kappa_t^{i,j} w^j(x^i|x_0, x_1) \right], \tag{44}$$

S106 · Original / 原文 · PDF p.20 · A5.Thmtheorem2.p1.4

where $\ell = \arg\min_{j\in[m]} \frac{\dot{\kappa}_t^{i,j}}{\kappa_t^{i,j}}$. Now by differentiating $p_t(x|x_0, x_1)$ we get

其中 $\ell = \arg\min_{j\in[m]}(\dot{\kappa}_t^{i,j}/\kappa_t^{i,j})$。对条件路径 $p_t(x|x_0, x_1)$ 求导得到下式。

$$\begin{aligned}
p_t(x|x_0, x_1) &= \prod_{i=1}^{N} p_t(x^i|x_0, x_1) \\
\dot{p}_t(x|x_0, x_1) &= \sum_{i=1}^{N} p_t(x^{\bar{i}}|x_0, x_1) \dot{p}_t(x^i|x_0, x_1) \\
&= \sum_{i=1}^{N} p_t(x^{\bar{i}}|x_0, x_1) \left[\sum_{j=1}^{m} \dot{\kappa}_t^{i,j} w^j(x^i|x_0, x_1) \right] \\
&= \sum_{i=1}^{N} p_t(x^{\bar{i}}|x_0, x_1) \left[\sum_{j\neq\ell} \dot{\kappa}_t^{i,j} w^j(x^i|x_0, x_1) + \dot{\kappa}_t^{i,\ell} w^\ell(x^i|x_0, x_1) \right] \\
&= \sum_{i=1}^{N} p_t(x^{\bar{i}}|x_0, x_1) \left[\sum_{j=1}^{m} \overbrace{\left[\dot{\kappa}_t^{i,j} - \kappa_t^{i,j} \frac{\dot{\kappa}_t^{i,\ell}}{\kappa_t^{i,\ell}} \right]}^{a_t^{i,j}} w^j(x^i|x_0, x_1) + \overbrace{\frac{\dot{\kappa}_t^{i,\ell}}{\kappa_t^{i,\ell}}}^{b_t^i} p_t(x^i|x_0, x_1) \right] \quad \text{[uses (44)]} \\
&= \sum_{i=1}^{N} \left[\sum_{j=1}^{m} a_t^{i,j} \overbrace{\left[\sum_z \delta_x(z^{\bar{i}}) p_t(z|x_0, x_1) \right]}^{=p_t(x^{\bar{i}}|x_0, x_1)} w^j(x^i|x_0, x_1) + b_t^i \overbrace{\left[\sum_z \delta_x(z^{\bar{i}}) \delta_x(z^i) p_t(z|x_0, x_1) \right]}^{=p_t(x|x_0, x_1)} \right] \\
&= \sum_z p_t(z|x_0, x_1) \sum_{i=1}^{N} \delta_x(z^{\bar{i}}) \overbrace{\left[\sum_{j=1}^{m} a_t^{i,j} w^j(x^i|x_0, x_1) + b_t^i \delta_x(z^i) \right]}^{u_t^i(x^i, z|x_0, x_1)} \quad \text{[normalization]} \\
&= -\mathrm{div}_x(p_t(\cdot|x_0, x_1) u_t(\cdot|x_0, x_1)),
\end{aligned}$$

S107 · Original / 原文 · PDF p.20 · A5.Thmtheorem2.p1.5

as required.

这正是所需结论。

E.4 Backward-time generating probability velocity.

E.4 反向时间生成概率速度

S108 · Original / 原文 · PDF p.20 · A5.SS4.p1.1

Here we prove the equivalent of Theorem 3 for backward-time generating probability field. But first, let us justify the backward sampling formula,

这里证明定理 3 对反向时间概率场的对应结论。先说明下列反向采样公式的合理性。

$$X^i_{t-h} \sim \delta_{X^i_t}(\cdot) - h u^i_t(\cdot, X_t). \tag{45}$$

S109 · Original / 原文 · PDF p.20 · A5.SS4.p1.2

Similar to (20) we have

类似式（20），有下式。

$$\mathbb{E}_{X_t} \prod_{i=1}^{N} \left[\delta_{X_t}(x^i) - h u^i_t(x^i, X_t)\right] = \mathbb{E}_{X_t}\left[\delta_{X_t}(x) - h \sum_{i=1}^{N} \delta_{X_t}(x^{\bar{i}}) u^i_t(x^i, X_t)\right] + o(h)$$

$$= p_t(x) + h \mathrm{div}_x(p_t u_t) + o(h) \overset{(16)}{=} p_t(x) - h\dot{p}_t(x) + o(h) = p_{t-h}(x) + o(h).$$

S110 · Original / 原文 · PDF p.20 · A5.SS4.p1.3

Therefore if the Continuity equation holds and $-u_t$ satisfies the conditions in (13) then given $X_t \sim p_t$, (45) provides an approximation $X_{t-h} \sim p_{t-h} + o(h)$. The change to the generating probability velocity in (22) to accommodate reverse time sampling is to replace the argmin in (41) with argmax,

所以若连续性方程成立且 $-u_t$ 满足式（13），则给定 $X_t \sim p_t$，式（45）近似产生 $X_{t-h} \sim p_{t-h} + o(h)$。要让式（22）的生成概率速度用于反向采样，只须将式（41）的 argmin 改为 argmax。

$$\ell = \ell(i,t) \triangleq \arg\max_{j \in [m]} \left[\dot{\kappa}^{i,j}_t / \kappa^{i,j}_t\right]. \tag{46}$$

S111 · Original / 原文 · PDF p.20 · A5.SS4.p1.4

An analogous result to Theorem 3 for backward-time sampling is therefore,

由此得到与定理 3 对应的反向时间采样结论。

Theorem 7 (Probability velocity of conditional paths, backward time). 定理 7（反向条件路径概率速度）

S112 · Original / 原文 · PDF p.21 · A5.Thmtheorem3.p1.1

The probability velocity $-u_t$, where u_t defined in (21) with $\ell = \arg\max_{j \in [m]} \left[\dot{\kappa}^{i,j}_t / \kappa^{i,j}_t\right]$ is a backward-time generating probability velocity for the conditional paths $p_t(x|x_0, x_1)$ defined in equations 7 and 8.

在式（21）中取 $\ell = \arg\max_{j \in [m]} (\dot{\kappa}^{i,j}_t / \kappa^{i,j}_t)$ 所得 u_t，其相反数 $-u_t$ 是式（7）、（8）定义的条件路径 $p_t(x|x_0, x_1)$ 的反向时间生成概率速度。

Proof (Theorem 7). 证明（定理 7）

S113 · Original / 原文 · PDF p.21 · A5.Thmtheorem4.p1.1

We follow the proof of Theorem 3 and indicate the relevant changes. First, for arbitrary $X_t \in \mathcal{D}$,

沿用定理 3 的证明，仅说明需要修改之处。首先，对任意 $X_t \in \mathcal{D}$，有下式。

$$\sum_{x^i} u_t^i(x^i, X_t) = 0, \tag{47}$$

S114 · Original / 原文 · PDF p.21 · A5.Thmtheorem4.p1.2

exactly using the same arguments as the forward-time case. Now, for $x^i \neq X_t^i$ we have

此处所用论证与前向时间完全相同。再对 $x^i \neq X_t^i$ 考察下式。

$$u_t^i(x^i, X_t|x_0, x_1) = \sum_{j=1}^{m} \left[\frac{\dot{\kappa}_t^{i,j}}{\kappa_t^{i,j}} - \frac{\dot{\kappa}_t^{i,\ell}}{\kappa_t^{i,\ell}}\right] \kappa_t^{i,j} w_t^j(x^i|x_0, x_1) \leq 0 \tag{48}$$

S115 · Original / 原文 · PDF p.21 · A5.Thmtheorem4.p1.3

due to ℓ being now the argmax of $\frac{\dot{\kappa}_t^{i,j}}{\kappa_t^{i,j}}$. Therefore $-u_t$ satisfies (13). Lastly, we notice that the proof of the Continuity Equation follows through exactly the same also in this case.

因为此时 ℓ 使 $\dot{\kappa}_t^{i,j}/\kappa_t^{i,j}$ 最大，所以 $-u_t$ 满足式（13）。最后，连续性方程的证明仍然逐步成立，不需其他改动。

E.5 Backward-time generating velocity for i.i.d. source $p(x_0)$ and simple paths

E.5 独立同分布源与简单路径的反向速度

S116 · Original / 原文 · PDF p.21 · A5.SS5.p1.1

Here we consider the case of probability paths defined via the conditionals in (9) with independent coupling $\pi(x_0, x_1) = p(x_0)q(x_1)$ and i.i.d. source distribution $p(x_0) = \prod_{i=1}^{N} p(x_0^i)$, where $p(x_0^i)$ is some PMF over $[d]$. In this case one can simplify the time-backward sampling formula in (25) by using the following one which is equivalent (i.e., their difference is divergence free and consequently generate the same probability path p_t):

这里考虑由式（9）条件分布定义的路径，使用独立耦合 $\pi(x_0, x_1) = p(x_0)q(x_1)$，且源分布独立同分布 $p(x_0) = \prod_i p(x_0^i)$，每个分量是 [d] 上某个 PMF。此时可用下列等价公式简化式（25）的反向采样；“等价”指二者的差是无散的，因此生成相同的概率路径 p_t。

$$\check{u}_t(x^i, X_t) = \frac{\dot{\kappa}_t}{\kappa_t}\left[\delta_{X_t}(x^i) - p(x^i)\right]. \tag{49}$$

S117 · Original / 原文 · PDF p.21 · A5.SS5.p1.2

The benefit in this equation is that it does not require the posterior $p_{0|t}$, which needs to be learned in general cases.

该公式的好处是不需要通常必须学习的后验 $p_{0|t}$。

S118 · Original / 原文 · PDF p.21 · A5.SS5.p2.1

To show that (49) is indeed a generating probability velocity it is enough to show that

要证明式（49）确实是生成概率速度，只需证明下式。

$$\mathrm{div}_x\left[p_t\left(\check{u}_t - \check{u}_t^\star\right)\right] = 0, \tag{50}$$

S119 · Original / 原文 · PDF p.21 · A5.SS5.p2.2

where $\check{u}_t^\star$ is the probability velocity in (25). Let us verify using (19):

其中 $\check{u}_t^\star$ 是式（25）的概率速度。使用式（19）验证如下。

$$
\begin{aligned}
\operatorname{div}_x\left[p_t\left(\check{u}_t-\check{u}_t^\star\right)\right] &= \sum_{i,z} p_t(z)\delta_z(x^{\bar{i}})\left[p(x^i)-\sum_{x_0,x_1}\delta_{x_0}(x^i)\frac{p_t(z|x_0,x_1)p(x_0)q(x_1)}{p_t(z)}\right] \quad \text{[normalization]}\\
&= \sum_{i,z}\delta_z(x^{\bar{i}})\left[p(x^i)p_t(z)-\sum_{x_0,x_1}\delta_{x_0}(x^i)p_t(z|x_0,x_1)p(x_0)q(x_1)\right]\\
&= \sum_{i,x_0,x_1}\left[p(x^i)-\delta_{x_0}(x^i)\right]\left(\sum_z \delta_z(x^{\bar{i}})p_t(z|x_0,x_1)\right)p(x_0)q(x_1)\\
&= \sum_{i,x_0,x_1}\left[p(x^i)-\delta_{x_0}(x^i)\right]p_t(x^{\bar{i}}|x_0,x_1)p(x_0^{\bar{i}})p(x_0^i)q(x_1)\\
&= \sum_{i,x_0^i,x_1}\left(\sum_{x_0^i}\left[p(x^i)p(x_0^i)-\delta_{x_0}(x^i)p(x_0^i)\right]\right)p_t(x^{\bar{i}}|x_0,x_1)p(x_0^{\bar{i}})q(x_1)\\
&= 0,
\end{aligned}
$$

S120 · Original / 原文 · PDF p.21 · A5.SS5.p2.3

where in the second to last equality we used the fact that the paths we are considering have the form: $p_t(x^{\bar{i}}|x_0,x_1)=\prod_{j\in[N]\setminus i}\left[\kappa_t\delta_{x_1}(x^j)+(1-\kappa_t)\delta_{x_0}(x^j)\right]$, and therefore do not depend on the i-th source token, x_0^i.

倒数第二个等号利用了此处路径的乘积形式：$p_t(x^{\bar{i}}|x_0,x_1)=\prod_{j\neq i}[\kappa_t\delta_{x_1}(x^j)+(1-\kappa_t)\delta_{x_0}(x^j)]$，因而不依赖源端第 i 个词元 x_0^i。

E.6 Corrector steps

E.6 校正步骤

Theorem 4.

S121 · Original / 原文 · PDF p.22 · A5.SS6.p1.2

For perfectly trained posteriors and $\alpha_t,\beta_t>0$, $t\in(0,1)$, $\bar{u}_t$ in (26) is a probability velocity, i.e., satisfies (13), and: (i) For $\alpha_t-\beta_t=1$, $\bar{u}_t$ provides a probability velocity generating p_t; (ii) For $\alpha_t-\beta_t=0$, repeatedly sampling with $\bar{u}_t$ at fixed $t\in(0,1)$ and sufficiently small h is guaranteed to converge to a sample from p_t.

重述定理 4：对于完全训练好的后验，若 α_t、$\beta_t>0$ 且 $t\in(0,1)$，式（26）的 $\bar{u}_t$ 满足式（13）；（i）$\alpha_t-\beta_t=1$ 时生成 p_t；（ii）$\alpha_t-\beta_t=0$ 时，在固定 t、足够小 h 下反复采样，保证收敛至服从 p_t 的样本。

Proof (Theorem 4). 证明（定理 4）

S122 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p1.1

First let us write explicitly $\bar{u}_t$ from (26):

首先将式（26）的 $\bar{u}_t$ 显式写成下式。

$$
\begin{aligned}
\bar{u}_t^i(x^i,X_t) &= \alpha_t\hat{u}_t^i(x^i,X_t)-\beta_t\check{u}_t^i(x^i,X_t)\\
&= \frac{\alpha_t\dot{\kappa}_t}{1-\kappa_t}p_{1|t}(x^i|X_t)+\frac{\beta_t\dot{\kappa}_t}{\kappa_t}p_{0|1}(x^i|X_t)-\left[\frac{\alpha_t\dot{\kappa}_t}{1-\kappa_t}+\frac{\beta_t\dot{\kappa}_t}{\kappa_t}\right]\delta_{X_t}(x^i).
\end{aligned}
\tag{51}
$$

S123 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p1.2

Since (51) is a sum of PMFs with coefficients that sum up to zero the first condition in (13), i.e., $\sum_{x^i}\bar{u}_t^i(x^i,X_t)=0$ holds. The second condition in (13) holds since for $t\in(0,1)$ we have $\frac{\alpha_t\dot{\kappa}_t}{1-\kappa_t},\frac{\beta_t\dot{\kappa}_t}{\kappa_t}\geq 0$. Now,

式（51）是若干 PMF 的线性组合，其系数之和为零，因此式（13）的第一条件 $\sum_{x^i}\bar{u}_t^i(x^i,X_t)=0$ 成立。第二条件也成立，因为 $t\in(0,1)$ 时，$\alpha_t\dot{\kappa}_t/(1-\kappa_t)$ 与 $\beta_t\dot{\kappa}_t/\kappa_t$ 都非负。接下来有下式。

$$
\operatorname{div}_x(p_t\bar{u}_t)=\alpha_t\operatorname{div}_x(p_t\hat{u}_t)-\beta_t\operatorname{div}_x(p_t\check{u}_t) \qquad \text{[normalization]}
$$

$$
\begin{aligned}
&= -\alpha_t \dot{p}_t(x) + \beta_t \dot{p}_t(x) && \text{[uses (16)]} \\
&= -(\alpha_t - \beta_t)\dot{p}_t(x). && (52)
\end{aligned}
$$

S124 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p2.1

For (i): Using (52) with $\alpha_t - \beta_t = 1$ we get that

对于（i），将 $\alpha_t-\beta_t=1$ 代入式（52），得到下式。

$$
\operatorname{div}_x(p_t \bar{u}_t) = -\dot{p}_t(x),
$$

i.e.

S125 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p2.3

, $\bar{u}_t$ satisfies the continuity equation and therefore generates p_t.

因此 $\bar{u}_t$ 满足连续性方程，进而生成 p_t。

S126 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p3.1

For (ii): Setting $\alpha_t - \beta_t = 0$ in (52) we get $\operatorname{div}_x(p_t \bar{u}_t) = 0$ and therefore similar to (20) we have

对于（ii），将 $\alpha_t-\beta_t=0$ 代入式（52），得到 $\operatorname{div}_x(p_t \bar{u}_t) = 0$，因此类似式（20）可得下式。

$$
\begin{aligned}
p_t(x) &= p_t(x) - h \operatorname{div}_x(p_t \bar{u}_t) \\
&= \mathbb{E}_{X_t}\left[\delta_{X_t}(x) + h \sum_{i=1}^{N} \delta_{X_t}(x^{\bar{i}}) \bar{u}_t^i(x^i, X_t)\right] \\
&= \sum_z p(x|z) p_t(z), && (53)
\end{aligned}
$$

S127 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p3.2

where using (51) we have

其中利用式（51），有下式。

$$
\begin{aligned}
p(x|z) = {} & h \sum_{i=1}^{N} \frac{\alpha_t \dot{\kappa}_t}{1-\kappa_t} \delta_z(x^{\bar{i}}) p_{1|t}(x^i|z) + h \sum_{i=1}^{N} \frac{\beta_t \dot{\kappa}_t}{\kappa_t} \delta_z(x^{\bar{i}}) p_{0|1}(x^i|z) \\
& + \left(1 - h \sum_{i=1}^{N} \left[\frac{\alpha_t \dot{\kappa}_t}{1-\kappa_t} + \frac{\beta_t \dot{\kappa}_t}{\kappa_t}\right]\right) \delta_z(x^i).
\end{aligned}
$$

S128 · Original / 原文 · PDF p.22 · A5.Thmtheorem5.p3.3

For sufficiently small $h > 0$ therefore $p(x|z)$ is a convex combination of PMFs (in red) x and consequently is itself a PMF in x, that is $p(x|z)$ is a probability transition matrix, and $p_t(x)$ is its stationary distribution, i.e., it is an eigenvector of $p(x|z)$ with eigenvalue 1, which is maximal. To prove convergence of the iterations in (53) we are left with showing that $p(x|z)$ is irreducible and a-periodic, see *Norris (1998)* (Theorem 1.8.3). Irreducibly of $p(x|z)$ can be shown by connecting each two states z, z' by changing one token at a time, and assuming that $p_{1|t}$ or $p_{0|t}$ are strictly positive (which is usually the case since as at-least one of them is defined as soft-max of finite logits); a-periodicity is proved by showing $p(x|x) > 0$ which is true as the coefficient of $\delta_z(x)$ is greater than zero for sufficiently small $h > 0$. Lastly, note that the iteration in (53) changes one token at a time. An approximation to this sampling can be achieved using our standard parallel sampling via (12), justified by (20).

当 h>0 足够小时，p(x|z) 是各个 PMF（原式红色部分）的凸组合，因而本身也是关于 x 的 PMF。于是 p(x|z) 是概率转移矩阵，$p_t(x)$ 为其平稳分布，即对应最大特征值 1 的特征向量。为证明式（53）的迭代收敛，还需证明转移矩阵不可约、非周期，见 Norris（1998）定理 1.8.3。若假设 $p_{1|t}$ 或 $p_{0|t}$ 严格为正，可通过每次改变一个词元连接任意状态 z、z′，从而说明不可约；通常至少一个后验由有限 logits 的 softmax 定

义，满足此正性。充分小 h 时 δz(x) 的系数大于零，所以 p(x|x)>0，证明非周期性。最后，式（53）每次只改变一个词元；可以用标准并行采样式（12）近似，式（20）为该近似提供依据。

E.7 Training

E.7 训练

Proposition 5.

S129 · Original / 原文 · PDF p.23 · A5.SS7.p1.2

The minimizer of $\mathcal{L}$ ((28)) is $\hat{w}_t^j(x^i|X_t)$ ((23)).

重述命题 5：损失 L（式 28）的极小化解是式（23）的后验 $\hat{w}_t^j(x^i|X_t)$。

Proof (Proposition 5). 证明（命题 5）

S130 · Original / 原文 · PDF p.23 · A5.Thmtheorem6.p1.1

It is enough to prove the claim for $m = 1$, with a single $w(x^i|x_0, x_1)$.

只需证明 m=1、即只有一个 $w(x^i|x_0, x_1)$ 时的结论。

$$
\begin{aligned}
\mathcal{L}(\theta) &= -\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_t \sum_{x_0,x_1,z,y^i} \log \hat{w}_t(y^i|z;\theta) w(y^i|x_0,x_1) p_t(z|x_0,x_1)\pi(x_0,x_1) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_t \sum_{z} p_t(z) \left[\sum_{y^i} \log \hat{w}_t(y^i|z;\theta) \left(\sum_{x_0,x_1} w(y^i|x_0,x_1)\frac{p_t(z|x_0,x_1)\pi(x_0,x_1)}{p_t(z)}\right)\right] \\
&= -\mathbb{E}_{t,X_t}\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{y^i} \log \hat{w}_t(y^i|X_t;\theta)\hat{w}_t(y^i|X_t)\right],
\end{aligned}
$$

S131 · Original / 原文 · PDF p.23 · A5.Thmtheorem6.p1.2

that amounts to minimizing the Cross Entropy loss between $\hat{w}_t(x^i|X_t;\theta)$ and $\hat{w}_t(x^i|X_t)$ for all $i \in [N]$, the minimizer of which satisfies $\hat{w}_t(x^i|X_t;\theta) \equiv \hat{w}_t(x^i|X_t)$.

上述计算等价于对所有 i 最小化 $\hat{w}_t(x^i|X_t;\theta)$ 与 $\hat{w}_t(x^i|X_t)$ 之间的交叉熵，极小化解满足 $\hat{w}_t(x^i|X_t;\theta) \equiv \hat{w}_t(x^i|X_t)$。

E.8 Time-independent posterior for masked modeling

E.8 掩码模型中与时间无关的后验

Proposition 6.

S132 · Original / 原文 · PDF p.23 · A5.SS8.p1.2

For paths defined by equations 7 and 9 with source $p(x) = \delta_m(x)$ the posterior $p_t(x_0, x_1|z) = p(x_0, x_1|z)$ is time-independent. Consequently, the probability denoiser $p_{1|t}(x^i|z) = p_1(x^i|z)$ is also time-independent.

重述命题 6：对式（7）、（9）定义且源分布为 $p(x) = \delta_M(x)$ 的路径，端点后验 $p_t(x_0, x_1|z) = p(x_0, x_1|z)$ 与时间无关；因此概率去噪器 $p_{1|t}(x^i|z) = p_1(x^i|z)$ 也与时间无关。

Proof (Proposition 6). 证明（命题 6）

S133 · Original / 原文 · PDF p.23 · A5.Thmtheorem7.p1.1

First,

首先有下式。

$$p_t(z^i|x_0,x_1) = (1-\kappa_t)\delta_m(z^i) + \kappa_t\delta_{x_1}(z^i) = \begin{cases}(1-\kappa_t) & z^i = m\\ \kappa_t\delta_{x_1}(z^i) & z^i \neq m\end{cases}$$

S134 · Original / 原文 · PDF p.23 · A5.Thmtheorem7.p1.2

and therefore

因此可得下式。

$$p_t(z|x_0,x_1) = \left[\prod_{i:z^i=m}(1-\kappa_t)\prod_{i:z^i\neq m}\kappa_t\right]\prod_{i:z^i\neq m}\delta_{x_1}(z^i).$$

S135 · Original / 原文 · PDF p.23 · A5.Thmtheorem7.p1.3

The posterior now gives

后验由此给出下式。

$$\begin{aligned}\frac{p_t(z|x_0,x_1)\pi(x_0,x_1)}{p_t(z)} &= \frac{\left[\prod_{i=1}^N p_t(z^i|x_0,x_1)\right]\pi(x_0,x_1)}{\sum_{\tilde{x}_0,\tilde{x}_1}\left[\prod_{j=1}^N p_t(z^j|\tilde{x}_0,\tilde{x}_1)\right]\pi(\tilde{x}_0,\tilde{x}_1)}\\ &= \frac{\cancel{\left[\prod_{i:z^i=m}(1-\kappa_t)\prod_{i:z^i\neq m}\kappa_t\right]}\left[\prod_{i:z^i\neq m}\delta_{x_1}(z^i)\right]\pi(x_0,x_1)}{\sum_{\tilde{x}_0,\tilde{x}_1}\cancel{\left[\prod_{j:z^j=m}(1-\kappa_t)\prod_{j:z^j\neq m}\kappa_t\right]}\left[\prod_{j:z^j\neq m}\delta_{\tilde{x}_1}(z^j)\right]\pi(\tilde{x}_0,\tilde{x}_1)}\\ &= p(x_0,x_1|z).\end{aligned}$$

S136 · Original / 原文 · PDF p.23 · A5.Thmtheorem7.p1.4

showing that the posterior is time-independent for dummy source distributions and convex paths. Consequently also the probability denoiser,

这表明，对 dummy 源分布和凸路径，后验与时间无关。因此概率去噪器也如下式。

$$p_{1|t}(x^i|z) = \sum_{x_0,x_1}\delta_{x_1}(x^i)\frac{p_t(z|x_0,x_1)\pi(x_0,x_1)}{p_t(z)} = \sum_{x_0,x_1}\delta_{x_1}(x^i)p(x_0,x_1|z),$$

S137 · Original / 原文 · PDF p.23 · A5.Thmtheorem7.p1.5

is time-independent.

同样与时间无关。

E.9 Continuous Flow Matching

E.9 连续流匹配

S138 · Original / 原文 · PDF p.24 · A5.SS9.p1.1

For completeness we provide the formulas for denoiser (x-prediction) and noise-prediction (ε-prediction) parameterizations of the generating velocity field $u : [0,1]\times\mathbb{R}^N \to \mathbb{R}^N$ appearing in Table 1.

为完整起见，这里给出表 1 中生成速度场 $u : [0,1]\times\mathbb{R}^N \to \mathbb{R}^N$ 的去噪器（x 预测）及噪声预测（ε 预测）参数化公式。

S139 · Original / 原文 · PDF p.24 · A5.SS9.p2.1

In Continuous Flow Matching one can chose several ways to define the probability paths *(Lipman et al., 2022; Liu et al., 2022; Albergo and Vanden-Eijnden, 2022; Pooladian et al., 2023; Tong et al., 2023)*:

连续流匹配中有若干定义概率路径的方式（Lipman 等，2022；Liu 等，2022；Albergo 与 Vanden-Eijnden，2022；Pooladian 等，2023；Tong 等，2023），如下列公式。

$$p_t(x) = \int p_t(x|x_0, x_1)\pi(x_0, x_1)dx_0dx_1 \tag{54}$$
$$= \int p_{1|t}(x|x_1)q(x_1)dx_1 \tag{55}$$
$$= \int p_{0|t}(x|x_0)p(x_0)dx_0. \tag{56}$$

Denoiser parameterization.

去噪器参数化

S140 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px1.p1.1

The conditional generating velocity field $u_t(x|x_1)$ for $p_t(x|x_1)$, i.e., satisfy the Continuity Equation 16, takes the form *(Lipman et al., 2022)*

生成条件路径 $p_t(x|x_1)$、即满足连续性方程（16）的条件速度场 $u_t(x|x_1)$ 具有下式形式（Lipman 等，2022）。

$$u_t(x|x_1) = \frac{\dot{\kappa}_t}{1-\kappa_t}(x_1 - x), \tag{57}$$

S141 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px1.p1.2

and the marginal generating velocity field is therefore given by the marginalization with the posterior $p_t(x_1|x)$,

因此，边缘生成速度场由后验 $p_t(x_1|x)$ 边缘化得到下式。

$$u_t(x) = \int \frac{\dot{\kappa}_t}{1-\kappa_t}(x_1 - x)\frac{p_{1|t}(x|x_1)q(x_1)}{p_t(x)}dx_1$$
$$= \frac{\dot{\kappa}_t}{1-\kappa_t}\left[\hat{x}_{1|t}(x) - x\right],$$

S142 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px1.p1.3

where

其中去噪器定义如下。

$$\hat{x}_{1|t}(x) = \int x_1 \frac{p_{1|t}(x|x_1)q(x_1)}{p_t(x)}dx_1 = \mathbb{E}_{X_1 \sim p_t(\cdot|x)}X_1. \tag{58}$$

S143 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px1.p1.4

This shows the continuous Flow Matching denoiser parameterization of the generating velocity field in Table 1.

这便得到表 1 中连续流匹配生成速度场的去噪器参数化。

Noise-prediction parameterization.

噪声预测参数化

S144 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px2.p1.1

The conditional generating velocity field for $p_t(x|x_0)$ takes the form

生成 $p_t(x|x_0)$ 的条件速度场具有下式形式。

$$u_t(x|x_0) = \frac{\dot{\kappa}_t}{\kappa_t}(x - x_0), \tag{59}$$

S145 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px2.p1.2

and the marginal generating velocity field in this case is given by marginalization with the posterior $p_t(x_0|x)$,

此时边缘生成速度由后验 $p_t(x_0|x)$ 边缘化得到下式。

$$\begin{aligned} u_t(x) &= \int \frac{\dot{\kappa}_t}{\kappa_t}(x - x_0)\frac{p_{0|t}(x|x_0)p(x_0)}{p_t(x)}dx_0 \\ &= \frac{\dot{\kappa}_t}{\kappa_t}\left[x - \hat{x}_{0|t}(x)\right], \end{aligned}$$

S146 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px2.p1.3

where

其中噪声预测器定义如下。

$$\hat{x}_{0|t}(x) = \int x_0 \frac{p_{0|t}(x|x_0)p(x_0)}{p_t(x)}dx_0 = \mathbb{E}_{X_0 \sim p_t(\cdot|x)} X_0. \tag{60}$$

S147 · Original / 原文 · PDF p.24 · A5.SS9.SSS0.Px2.p1.4

This shows the continuous Flow Matching noise-prediction parameterization of the generating velocity field in Table 1.

这便得到表 1 中连续流匹配生成速度场的噪声预测参数化。

E.10 Scheduler change formula

E.10 调度器变换公式

Proposition 8. 命题 8

S148 · Original / 原文 · PDF p.24 · A5.Thmtheorem8.p1.1

Assume a conditional probability path as in (9), then for any two schedulers κ_t, κ'_t, and $\hat{w}_t(x^i|z), \hat{w}'_t(x^i|z)$ their corresponding posteriors as in (23),

假设条件概率路径为式（9），则任意两个调度器 κ_t、κ'_t，以及其对应的式（23）后验 $\hat{w}_t(x^i|z)$、$\hat{w}'_t(x^i|z)$，满足下式。

$$\hat{w}_{t'}(x^i|z) = \hat{w}'_t(x^i|z), \tag{61}$$

S149 · Original / 原文 · PDF p.24 · A5.Thmtheorem8.p1.2

where $t' = \kappa^{-1}_{\kappa'_t}$, and κ^{-1} is the inverse of κ.

其中 $t' = \kappa^{-1}(\kappa'_t)$，$\kappa^{-1}$ 是 κ 的逆函数。

Proof (Proposition 8). 证明（命题 8）

S150 · Original / 原文 · PDF p.25 · A5.Thmtheorem9.p1.1

For a conditional probability path as in (9),

对于式（9）的条件概率路径，有下式。

$$\begin{aligned} p_{t'}(x^i|x_0, x_1) &= \prod_{i=1}^{N} p_{t'}(x^i|x_0, x_1) && (62) \\ &= \prod_{i=1}^{N} \left[(1 - \kappa_{t'})\delta_{x_0}(x^i) + \kappa_{t'}\delta_{x_1}(x^i)\right] && (63) \end{aligned}$$

$$= \prod_{i=1}^{N} \left[(1-\kappa'_t)\delta_{x_0}(x^i) + \kappa'_t \delta_{x_1}(x^i)\right] \tag{64}$$

$$= \prod_{i=1}^{N} p'_t(x^i|x_0, x_1) \tag{65}$$

$$= p'_t(x^i|x_0, x_1), \tag{66}$$

S151 · Original / 原文 · PDF p.25 · A5.Thmtheorem9.p1.2

where in the 3rd equality we used $\kappa_{t'} = \kappa'_t$. Thus, also for the marginal probability path as in (7),

第三个等号使用 $\kappa_{t'} = \kappa'_t$。因此，式（7）的边缘概率路径也满足下式。

$$p_{t'}(x) = \sum_{x_0, x_1 \in \mathcal{D}} p_{t'}(x|x_0, x_1)\pi(x_0, x_1) \tag{67}$$

$$= \sum_{x_0, x_1 \in \mathcal{D}} p'_t(x|x_0, x_1)\pi(x_0, x_1) \tag{68}$$

$$= p'_t(x), \tag{69}$$

S152 · Original / 原文 · PDF p.25 · A5.Thmtheorem9.p1.3

where in the 2nd equality we used $p_{t'}(x|x_0, x_1) = p'_t(x|x_0, x_1)$. Finally the change of scheduler for a posterior as defined in (23),

第二个等号使用 $p_{t'}(x|x_0, x_1) = p'_t(x|x_0, x_1)$。最后，对式（23）的后验进行调度器变换得到下式。

$$\hat{w}_{t'}(x^i|z) = \sum_{x_0, x_1 \in \mathcal{D}} w(x^i|x_0, x_1) p_{t'}(x_0, x_1|z) \tag{70}$$

$$= \sum_{x_0, x_1 \in \mathcal{D}} w(x^i|x_0, x_1) \frac{p_{t'}(z|x_0, x_1)\pi(x_0, x_1)}{p_{t'}(z)} \tag{71}$$

$$= \sum_{x_0, x_1 \in \mathcal{D}} w(x^i|x_0, x_1) \frac{p'_t(z|x_0, x_1)\pi(x_0, x_1)}{p'_t(z)} \tag{72}$$

$$= \sum_{x_0, x_1 \in \mathcal{D}} w(x^i|x_0, x_1) p'_t(x_0, x_1|z) \tag{73}$$

$$= \hat{w}'_t(x^i|z) \tag{74}$$

S153 · Original / 原文 · PDF p.25 · A5.Thmtheorem9.p1.4

where in the 3rd equality we used both $p_{t'}(z|x_0, x_1) = p'_t(z|x_0, x_1)$ and $p_{t'}(z) = p'_t(z)$.

第三个等号同时使用 $p_{t'}(z|x_0, x_1) = p'_t(z|x_0, x_1)$ 与 $p_{t'}(z) = p'_t(z)$。

Appendix F Inference time

附录 F 推断时间

S154 · Original / 原文 · PDF p.25 · A6.p1.1

One potential benefit of non-autoregressive decoding is improved latency due to a significantly lower number of decoding steps. To demonstrate that, we measure the average latency of the proposed method compared with the autoregressive alternative using a single A100 GPU with 80 GB of RAM. We calculate the average latency time on the HumanEval benchmark using a batch size of 1. When considering 256 NFEs, the proposed method was found to be ∼2.5x faster than the autoregressive model (19.97 vs. 50.94 seconds on average per example). However, when considering 512 NFEs, both methods reach roughly the same latency. These results make sense as the number of tokens in most of the examples in HumanEval are below 512. Notice, that these results analyze latency and not model throughput. Due to the kv-caching mechanism following the autoregressive approach will result in significantly better throughput compared to the proposed approach *Ziv et al. (2024)*. We leave the construction of a kv-cache mechanism to the proposed approach for future research.

非自回归解码的潜在优点是解码步数明显更少，从而减少延迟。作者使用一块 80 GB A100 GPU，在 HumanEval、批量大小 1 上比较平均延迟。256 NFE 时，本文方法比自回归模型快约 2.5 倍：每例平均 19.97 对 50.94 秒；512 NFE 时，二者延迟大致相同。这与 HumanEval 多数例子的词元数少于 512 相符。这里分析的是延迟而非吞吐量；由于 KV-cache 机制，自回归方案的吞吐量会明显更好（Ziv 等，2024）。为本文方法构建 KV-cache 留待后续研究。译注：这是论文报告的 GPU 实验，本次整理未运行该实验。

Appendix G Experimental setup

附录 G 实验设置

G.1 Text

G.1 文本

Data.

数据

S155 · Original / 原文 · PDF p.26 · A7.SS1.SSS0.Px1.p1.1

We use three splits of data. First is OpenWebText *(Gokaslan and Cohen, 2019)*. Second is the same mix used in Llama-2 *(Touvron et al., 2023)*, including textual and code data. For the code-focused models we use the same split used in CodeLlama *(Roziere et al., 2023)*. For the small models, we use OpenWebText. For the big models we use the Llama-2 and CodeLlama mixes.

我们使用三类数据。第一类是 OpenWebText（Gokaslan 与 Cohen，2019）；第二类是 Llama-2（Touvron 等，2023）使用的相同混合数据，包括文本与代码；偏重代码的模型使用与 CodeLlama（Roziere 等，2023）相同的数据划分。小模型用 OpenWebText，大模型用 Llama-2 与 CodeLlama 混合数据。

Models.

模型

S156 · Original / 原文 · PDF p.26 · A7.SS1.SSS0.Px2.p1.1

We train two sizes of models: small (150M parameters) and large (1.7B parameters). For the small model we used a DiT transformer architecture *(Peebles and Xie, 2022)* with 12 layers, 12 attention heads, and hidden dimension of 768. We also used GPT2 tokenizer. The small models were trained on OpenWebText. For the large model, we use also used a DiT transformer architecture but with 48 layers, 24 attention heads, and hidden dimension of 1536 *(Peebles and Xie, 2022)*. For these models we used a tiktoken tokenizer. The large models were trained on the Llama-2 mix and the CodeLlama mix. For both models we used ROPE *(Su et al., 2024)* embedding with $\theta = 10000$. Models are trained with Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use dropout rate of 0.1. Models are trained with a warm-up of 2500 steps, with a peak learning rate of 3e-4. We train the big models with batch size of 4096 for 1.3 million iterations and the big models with batch size of 512 for 400 thousand iterations.

训练两种规模：小模型 1.5 亿参数、大模型 17 亿参数。小模型为 DiT Transformer（Peebles 与 Xie，2022），有 12 层、12 个注意力头、隐藏维度 768，使用 GPT2 tokenizer，在 OpenWebText 训练。大模型也采用 DiT，但有 48 层、24 个头、隐藏维度 1536，使用 tiktoken，在 Llama-2 与 CodeLlama 混合数据训练。两者均用 θ=10000 的 ROPE 嵌入（Su 等，2024），Adam 的 β_1=0.9、β_2=0.999，dropout=0.1，预热 2500 步，峰值学习率 3×10^{-4}。原文末句先写“大模型以批量 4096 训练 130 万次迭代”，又写“大模型以批量 512 训练 40 万次迭代”；两处都写 big models，未自行改成 small。

Entropy metric.

熵指标

S157 · Original / 原文 · PDF p.26 · A7.SS1.SSS0.Px3.p1.1

We report the entropy of tokens within a sequence, averaged over all generated sequences. This intuitively quantifies the diversity of tokens within a given sequence. It's important to note that when computing sequence entropy, tokens not present in the sequence are excluded from consideration.

我们报告每个序列内部词元熵，再对所有生成序列取平均。直观上，它度量单个序列内的词元多样性。计算序列熵时，不计入该序列中未出现的词元。

Generative perplexity metric.

生成困惑度指标

S158 · Original / 原文 · PDF p.26 · A7.SS1.SSS0.Px4.p1.1

The generative perplexity metric is the average likelihood of generated text evaluated with a second (usually stronger) model. We report the generative perplexity when averaged over 1000 samples.

生成困惑度指标是用第二个（通常更强）模型评估生成文本的平均似然。本文报告对 1000 个样本平均的生成困惑度。

Double precision sampling.

双精度采样

S159 · Original / 原文 · PDF p.26 · A7.SS1.SSS0.Px5.p1.1

Zheng et al. (2024) demonstrated that sampling from a high-dimensional distribution with full precision can lead to a similar affect as sampling with temperature. We evaluate our model using a categorical sampler in double precision. Table 5 presents the results of baselines compared to our method.

Zheng 等（2024）表明，从高维分布全精度采样可能产生类似温度采样的效果。我们以双精度类别采样器评估模型，表 5 比较本文方法与基线。

METHOD	NFE	LLAMA-2↓	LLAMA-3↓	GPT2↓	ENTROPY
Data	-	7.0	9.4	14.7	7.7
Autoregressive	1024	31.4	54.8	45.3	7.1
Savinov et al. (2021)	200	29.5	45.1	34.7	5.2
Austin et al. (2021a)	1000	697.6	768.8	837.8	7.6
Han et al. (2022)	>10000	73.3	203.1	99.2	4.8
Lou et al. (2023)	256 / 512 / 1024	56.6 / 54.0 / 56.1	122.1 / 115.7 / 117.7	115.0 / 107.8 / 109.5	8.1 / 8.1 / 8.1
Campbell et al. (2024)	256 / 512 / 1024	52.0 / 54.6 / 50.9	106.0 / 114.1 / 102.9	102.6 / 107.1 / 103.4	8.0 / 8.1 / 8.0
FM (equation 9)	256 / 512 / 1024	51.3 / 53.3 / 50.1	104.0 / 115.0 / 101.3	100.8 / 107.4 / 97.5	8.0 / 8.1 / 8.0
FM (equation 10)	256 / 512 / 1024	51.9 / 52.7 / 50.0	104.7 / 113.9 / 100.5	99.2 / 105.1 / 95.8	8.0 / 8.1 / 8.0

源图 / 表 / 代码：英文 PDF 第 26 页 · A7.T5

Table 5: Double precision sampling. Generative perplexity on unconditional text generation compared to prior work. All models are sampled without the use of temperature or corrector steps.

表 5：双精度采样的无条件文本生成困惑度比较；所有模型均不使用温度或校正步骤。

表头：Method = 方法，Data = 数据，NFE = 函数评估次数，Entropy = 熵，Pass@k = k 次候选内至少一项通过的比例；向下箭头为越低越好，向上箭头为越高越好。Autoregressive = 自回归，Oracle length = 给定真实解答长度。数值按原表保留，斜杠分隔值不合并。

G.2 Image

G.2 图像

Models.

模型

S160 · Original / 原文 · PDF p.26 · A7.SS2.SSS0.Px1.p1.1

For all our experiments on CIFAR10 we use the U-Net architecture as in *Dhariwal and Nichol (2021)*, with following three changes to make it fully discrete and time independent (as we used mask modeling): (i) We replace the first layer with an embedding table of size 257×96, and we stack the channel features such that the input to the U-Net is of shape $288 \times 32 \times 32$. (ii) We enlarge the size of the final layer to output a tensor of shape $3 \times 32 \times 32 \times 257$. (iii) We remove the time dependency from architecture. The hyper-parameters of the architecture: channels 96 , depth 5, channels multiple [3,4,4], heads channels 64, attention resolution 16, dropout 0.4, which gives a total parameters count of 113M. We optimize the network using Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a learning rate of 1e-4. We trained with an effective batch size pf 512 for roughly 300K iterations.

所有 CIFAR10 实验采用 Dhariwal 与 Nichol（2021）的 U-Net，并做三项改动，使其完全离散且与时间无关（使用掩码建模）：（i）首层换成 257×96 的嵌入表，堆叠通道特征，使输入形状为 288×32×32；（ii）末层增大为输出 3×32×32×257 的张量；（iii）移除架构的时间依赖。架构超参数为通道数 96、深度 5、通道倍数 [3,4,4]、每头通道数 64、注意力分辨率 16、dropout 0.4，总参数 1.13 亿。Adam 使用 β_1=0.9、β_2=0.999、学习率 10^{-4}，有效批量 512，训练约 30 万次迭代。

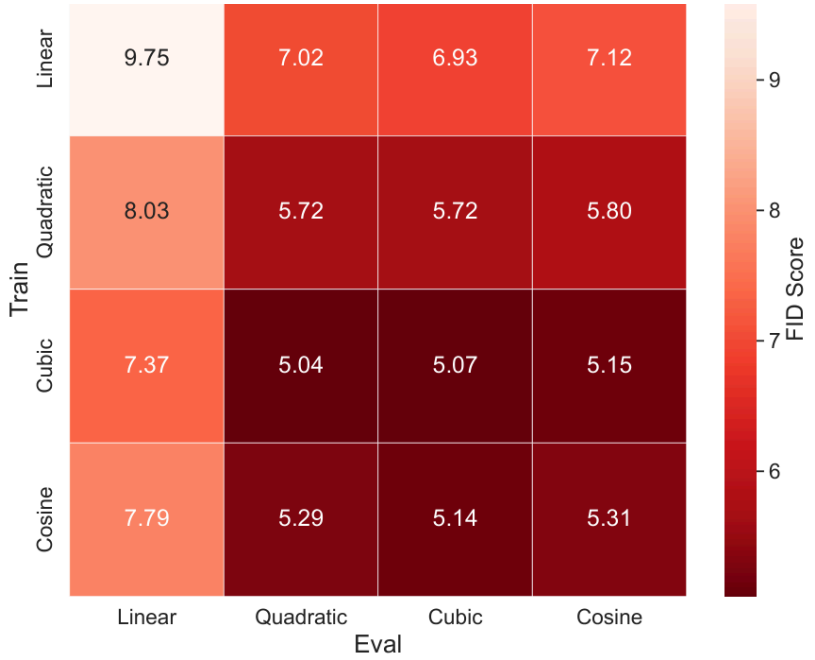
Scheduler ablation.

调度器消融

S161 · Original / 原文 · PDF p.27 · A7.SS2.SSS0.Px2.p1.1

Figure 8 shows FID of our method with four different schedulers: Linear, Quadratic, Cubic, Cosine, both for training and evaluation. That is, for each scheduler we trained a model and evaluate FID with all four schedulers. We observe a high variance in FID between different schedulers, with the Cubic scheduler generally performing the best on both training and evaluation.

图 8 比较线性、二次、三次、余弦四种调度器在训练与评估时的 FID；对每种训练调度器训练一个模型，再用全部四种调度器评估。不同调度器之间的 FID 变化很大，三次调度器总体在训练、评估中均最好。



源图 / 表 / 代码：英文 PDF 第 27 页 · A7.F8

Figure 8: Comparison of FID on CIFAR10 with four schedulers: Linear, Quadratic, Cubic, Cosine, for both train and evaluation. Corrector sampling is not used in this experiment (i.e., $\alpha_t = 0$ in (26)), and temperature is set to 1.

图 8：CIFAR10 上四种调度器（线性、二次、三次、余弦）在训练与评估组合下的 FID。此实验不使用校正采样（原文写式 26 的 α_t=0），温度为 1。

Comparison with baselines.

基线比较

S162 · Original / 原文 · PDF p.27 · A7.SS2.SSS0.Px3.p1.1

In the following, we provide implementation details for producing Figure 3(a), that compares our schedulers and sampling algorithm with those employed by previous works.

下面给出图 3(a) 的实现细节，该图比较本文调度器、采样算法与先前方法的设置。

Cubic Scheduler (Ours).

三次调度器（本文）

S163 · Original / 原文 · PDF p.27 · A7.SS2.SSS0.Px4.p1.1

For the Cubic scheduler we set the corrector scheduler as above to,

对于三次调度器，校正调度器按前述方式设置如下。

$$\alpha_t = 1 + \alpha t^a (1-t)^b, \quad \beta_t = \alpha_t - 1, \tag{75}$$

S164 · Original / 原文 · PDF p.27 · A7.SS2.SSS0.Px4.p1.2

and we search over the parameters $a, b \in \{0, 0.25, 0.5, 1, 2, 2.5, 3\}$, and $\alpha \in \{6, 8, 10, 12, 14\}$. Additionally, we search over the temperature $\in \{1, 0.9, 0.8\}$. We find that $a = 2$, $b = 0.25$, $\alpha = 12$ give best FID.

搜索 $a, b \in \{0, 0.25, 0.5, 1, 2, 2.5, 3\}$ 及 $\alpha \in \{6, 8, 10, 12, 14\}$，同时搜索温度 {1,0.9,0.8}。发现 a=2、b=0.25、α=12 时 FID 最佳。

Linear Scheduler *(Campbell et al., 2024)*.

线性调度器（Campbell 等，2024）

S165 · Original / 原文 · PDF p.27 · A7.SS2.SSS0.Px5.p1.1

For the linear scheduler we search over two additional hyper-parameters of the method: (i) For corrector scheduler as in (26), we set $\alpha_t = 1 + t\eta$, $\beta_t = \alpha_t - 1$, where η is the stochasticity parameter as in *Campbell et al. (2024)*, and search over $\eta \in \{0, 1, 2, 5, 10, 15\}$. (ii) We search over temperature in $\{1, 0.9, 0.8\}$. Finally, we find that the best FID is a achieved by $\eta = 10$ and temperature 0.9.

对于线性调度器，另外搜索两类超参数：（i）式（26）的校正调度器设为 $\alpha_t=1+t\eta$、$\beta_t=\alpha_t-1$，η 是 Campbell 等（2024）的随机性参数，搜索 η∈{0,1,2,5,10,15}；（ii）搜索温度 {1,0.9,0.8}。最终 η=10、温度 0.9 时 FID 最佳。

MaskGIT *(Chang et al., 2022)*.

MaskGIT（Chang 等，2022）

S166 · Original / 原文 · PDF p.27 · A7.SS2.SSS0.Px6.p1.1

For the MaskGIT we train and sample with the Cosine scheduler $\kappa(t) = 1 - \cos\left(\frac{\pi}{2}t\right)$ which is reported to achieve best results by *Chang et al. (2022)*. For sampling we adjust the code from the re-implementation of *Besnier and Chen (2023)*. In addition, we also search over the temperature in $\{1, 0.9, 0.8, 0.7, 0.6, 0.5\}$, and we find the best FID is achieved by temperature 1.

MaskGIT 的训练与采样采用余弦调度器 $\kappa(t) = 1 - \cos(\pi t/2)$，Chang 等（2022）报告它表现最佳。采样代码由 Besnier 与 Chen（2023）的复现修改而来。我们还搜索温度 {1,0.9,0.8,0.7,0.6,0.5}，发现温度 1 时 FID 最佳。

Appendix H Code generation - qualitative examples
附录 H 代码生成定性例子

H.1 Success cases
H.1 成功案例

```
from typing import List


def below_zero(operations: List[int]) -> bool:
    """ You're given a list of deposit and withdrawal operations
    on a bank account that starts with
    zero balance. Your task is to detect if at any point the balance
    of account fallls below zero, and
    at that point function should return True.
    Otherwise it should return False.
    >>> below_zero([1, 2, 3])
    False
    >>> below_zero([1, 2, -4, 5])
    True
    """
```

```
    balance = 0
    for operation in operations:
        if balance + operation < 0:
            return True
        balance = balance + operation
    return False
```

源图 / 表 / 代码：英文 PDF 第 28 页 · A8.SS1.fig1

函数说明：账户余额从零开始，依次执行给定存取款操作；只要任一时刻余额低于零立即返回 True，否则返回 False。示例 below_zero([1,2,3]) 为 False；below_zero([1,2,-4,5]) 为 True。

```
def encode_shift(s: str):
    """
    returns encoded string by shifting every character by 5 in the alphabet.
    """
    return "".join([chr(((ord(ch) + 5 - ord("a")) % 26) + ord("a")) for ch in s])


def decode_shift(s: str):
    """
    takes as input string encoded with encode_shift function.
    Returns decoded string.
    """
```

```
    decoded = []
    for ch in s:
        decoded.append(chr(((ord(ch) - 5 - ord("a")) % 26) + ord("a")))
    return "".join(decoded)
```

源图 / 表 / 代码：英文 PDF 第 28 页 · A8.SS1.fig2

encode_shift：把每个字母沿字母表移动 5 位，返回编码字符串。decode_shift：输入由 encode_shift 编码的字符串，返回解码结果。原 PDF 中的取模 % 26 保持不变。

```
from typing import List, Any


def filter_integers(values: List[Any]) -> List[int]:
    """ Filter given list of any python values only for integers
    >>> filter_integers(['a', 3.14, 5])
    [5]
    >>> filter_integers([1, 2, 3, 'abc', {}, []])
    [1, 2, 3]
    """
```

```
    return [value for value in values if isinstance(value, int)]
```

源图 / 表 / 代码：英文 PDF 第 29 页 · A8.SS1.fig3

函数说明：给定包含任意 Python 值的列表，只筛选整数。示例 [a,3.14,5] 得 [5]；[1,2,3,abc,{},[]] 得 [1,2,3]。测试中的字符串仍以原代码引号为准。

```
def common(l1: list, l2: list):
    """Return sorted unique common elements for two lists.
    >>> common([1, 4, 3, 34, 653, 2, 5], [5, 7, 1, 5, 9, 653, 121])
    [1, 5, 653]
    >>> common([5, 3, 2, 8], [3, 2])
    [2, 3]

    """
```

```
    return sorted(set(l1) & set(l2))
```

源图 / 表 / 代码：英文 PDF 第 29 页 · A8.SS1.fig4

函数说明：返回两个列表中共有的、去重且排序后的元素。示例 common([1,4,3,34,653,2,5],[5,7,1,5,9,653,121]) 得 [1,5,653]；common([5,3,2,8],[3,2]) 得 [2,3]。

```
def fibfib(n: int):
    """The FibFib number sequence is a sequence similar to
    the Fibbonacci sequnece that's defined as follows:
    fibfib(0) == 0
    fibfib(1) == 0
    fibfib(2) == 1
    fibfib(n) == fibfib(n-1) + fibfib(n-2) + fibfib(n-3).
    Please write a function to efficiently compute the
    n-th element of the fibfib number sequence.
    >>> fibfib(1)
    0
    >>> fibfib(5)
    4
    >>> fibfib(8)
    24
    """
```

```
    if n == 0:
        return 0
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibfib(n-1) + fibfib(n-2) + fibfib(n-3)
```

源图 / 表 / 代码：英文 PDF 第 29 页 · A8.SS1.fig5

函数说明：FibFib 类似 Fibonacci，定义 fibfib(0)=0、fibfib(1)=0、fibfib(2)=1，且 fibfib(n)=fibfib(n−1)+fibfib(n−2)+fibfib(n−3)。请高效计算第 n 项。示例 fibfib(1)=0、fibfib(5)=4、fibfib(8)=24。

H.2 Failure cases

H.2 失败案例

```
def solve(s):
    """You are given a string s.
    if s[i] is a letter, reverse its case from lower to upper or vise versa,
    otherwise keep it as it is.
    If the string contains no letters, reverse the string.
    The function should return the resulted string.
    Examples
    solve("1234") = "4321"
    solve("ab") = "AB"
    solve("#a@C") = "#A@c"
    """
```

```
    if s is None:
        return ""
    if len(s) == 0:
        return s
    # result string
    res = str()
    # to lower case
    s = s.lower()
    for i in range(len(s)):
        ch = s[i]
        if ch.islower():
            res.append(ch)
        elif ch.isupper():
            res.append(ch.lower())
        else:
            res.append(ch)
    return res
```

源图 / 表 / 代码：英文 PDF 第 30 页 · A8.SS2.fig1

函数说明：给定字符串 s，若 s[i] 是字母，则大小写互换，否则原样保留；若完全没有字母，则反转字符串，返回结果。例 solve("1234")="4321"、solve("ab")="AB"、solve("#a@C")="#A@c"。代码注释：result string＝结果字符串；to lower case＝转为小写。黄色输出是失败案例，保留其实现错误。

```
from typing import List, Tuple

def rolling_max(numbers: List[int]) -> List[int]:
    """ From a given list of integers, generate a list of rolling maximum element
    found until given moment in the sequence.
    >>> rolling_max([1, 2, 3, 2, 3, 4, 2])
    [1, 2, 3, 3, 3, 4, 4]
    """
```

```
    max_number = 0
    result = []
    for number in numbers:
        max_number = max(result)
        if number > max_number:
            result.append(number)
        else:
            result = [max_number]
    return result
```

源图 / 表 / 代码：英文 PDF 第 30 页 · A8.SS2.fig2

函数说明：从整数列表生成逐位置的历史最大值列表。例 rolling_max([1,2,3,2,3,4,2]) 应得 [1,2,3,3,3,4,4]。黄色代码保留失败输出。

```
def truncate_number(number: float) -> float:
    """ Given a positive floating point number, it can be decomposed into
    and integer part (largest integer smaller than given number) and decimals
    (leftover part always smaller than 1).

    Return the decimal part of the number.
    >>> truncate_number(3.5)
    0.5
    """
```

```
    return math.floor(number)
```

源图 / 表 / 代码：英文 PDF 第 31 页 · A8.SS2.fig3

函数说明：正浮点数可以分成整数部分（小于给定数的最大整数）与小数部分（剩余部分，总小于 1）；返回小数部分。例 truncate_number(3.5)=0.5。原生成代码返回 floor，按失败样例保留。

```python
from typing import List, Tuple

def sum_product(numbers: List[int]) -> Tuple[int, int]:
    """ For a given list of integers, return a tuple consisting of a sum and a
    product of all the integers in a list. Empty sum should be equal to 0 and
    empty product should be equal to 1.
    >>> sum_product([])
    (0, 1)
    >>> sum_product([1, 2, 3, 4])
    (10, 24)
    """
```

```python
    if len(numbers) == 0 or numbers[0] == 0:
        return 0, 1
    sum = 0
    product = 1
    for num in numbers:
        if abs(num) != 0:
            product = product * num
            sum += num
    return sum, product
```

源图 / 表 / 代码：英文 PDF 第 31 页 · A8.SS2.fig4

函数说明：给定整数列表，返回全部整数之和与积组成的元组；空和为 0，空积为 1。例 sum_product([])=(0,1)，sum_product([1,2,3,4])=(10,24)。

```python
from typing import List

def factorize(n: int) -> List[int]:
    """ Return list of prime factors of given integer in the order from smallest
    to largest. Each of the factors should be listed number of times corresponding
    to how many times it appeares in factorization. Input number should be equal
    to the product of all factors.
    >>> factorize(8)
    [2, 2, 2]
    >>> factorize(25)
    [5, 5]
    >>> factorize(70)
    [2, 5, 7]
    """
```

```python
    factors = []
    factor = 1
    for i in range(2, n):
        factor *= i
        while n % factor == 0:
            factors.append(factor)
            factor += 1
    return factors
```

源图 / 表 / 代码：英文 PDF 第 31 页 · A8.SS2.fig5

函数说明：将给定整数分解为质因数，按从小到大返回列表；每个因数按其分解重数重复，所有因数之积应等于输入。例 factorize(8)=[2,2,2]，factorize(25)=[5,5]，factorize(70)=[2,5,7]。

```python
from typing import List, Optional

def longest(strings: List[str]) -> Optional[str]:
    """ Out of list of strings, return the longest one. Return the first one in
     case of multiple strings of the same length. Return None in case the input
     list is empty.
     >>> longest([])

     >>> longest(['a', 'b', 'c'])
     'a'
     >>> longest(['a', 'bb', 'ccc'])
     'ccc'
     """
    long = ""
    for string in strings:
        if len(string) > len(long):
            long = string
    return long
```

源图 / 表 / 代码：英文 PDF 第 32 页 · A8.SS2.fig6

函数说明：返回字符串列表中最长者；并列时返回第一个；空列表返回 None。例 longest([]) 无值，longest([a,b,c]) 得 a，longest([a,bb,ccc]) 得 ccc；字符串按原代码引号理解。

H.3 Infilling

H.3 代码填空

```
def bubbleSort(arr):
    n = len(arr)
    # optimize code, so if the array is already sorted, it doesn't need
    # to go through the entire process
    # Traverse through all array elements
    for i in range(n-1):

        # range(n) also work but outer loop will
        # repeat one time more than needed.
        # Last i elements are already in place
        swapped = False
        for j in range(0, n-i-1):
            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j + 1]:
                swapped = True
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
        if not swapped:
            # if we haven't needed to make a single swap, we
            # can just exit the main loop.
            return
```

源图 / 表 / 代码：英文 PDF 第 33 页 · A8.SS3.fig1

代码注释逐项：优化算法，使已排序数组无需走完整过程；遍历全部数组元素；range(n) 也能用，但外循环会多执行一次；最后 i 个元素已就位；遍历下标 0 到 n−i−1；若当前元素大于下一个则交换；若一次交换都不需要，可直接退出主循环。黄色为填空生成片段。

```
# Function to perform Breadth First Search on a graph
# represented using adjacency list
def bfs(adjList, source , visited):
    # Create a queue for BFS
    q = deque()

    # Mark the current node as visited and enqueue it
    visited[ source ] = True

    q.append( source )

    # Iterate over the queue
    while q:
        # Dequeue a vertex from queue and print it
        currentNode = q.popleft()
        print( currentNode , end=" ")

        # Get all adjacent vertices of the dequeued vertex
        # If an adjacent has not been visited, then mark it visited and enqueue it
        for  adjacent  in adjList[ currentNode ]:

            if not visited[ adjacent ]:

                visited[ adjacent ] = True

                q.append( adjacent )
```

源图 / 表 / 代码：英文 PDF 第 34 页 · A8.SS3.fig2

代码注释逐项：对邻接表表示的图做广度优先搜索；建立 BFS 队列；标记当前节点已访问并入队；遍历队列；出队一个顶点并打印；获取该顶点全部相邻顶点；未访问的邻点标记后入队。

```python
# Returns index of x in arr if present, else -1
def binary_search(arr, low, high, x):

    # Check base case
    if high >= low:

        mid = (high + low) // 2

        # If element is present at the middle itself
        if arr[mid] == x:
            return mid

        # If element is smaller than mid, then it can only
        # be present in left subarray
        elif arr[mid] > x:

            return binary_search(arr, low, mid - 1, x)

        # Else the element can only be present in right subarray
        else:
            return binary_search(arr, mid + 1, high, x)

    else:
        # Element is not present in the array
        return -1
```

源图 / 表 / 代码：英文 PDF 第 34 页 · A8.SS3.fig3

代码注释逐项：若 x 存在于 arr，返回其下标，否则 −1；检查基本情形；若元素就在中间直接返回；若元素小于中间值，则只可能在左半数组；否则只可能在右半数组；若数组中不存在该元素，则返回 −1。

```python
# Python program for Dijkstra's single
# source shortest path algorithm. The program is
# for adjacency matrix representation of the graph
class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex \t Distance from Source")
        for node in range(self.V):
            print(node, "\t\t ", dist[node])

    # A utility function to find the vertex with
    # minimum distance value, from the set of vertices
    # not yet included in shortest path tree
    def minDistance(self, dist, sptSet):

        # Initialize minimum distance for next node
        min = 1e7

        # Search not nearest vertex not in the
        # shortest path tree
        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False:
                min = dist[v]
                min_index = v

        return min_index

    # Function that implements Dijkstra's single source
    # shortest path algorithm for a graph represented
    # using adjacency matrix representation
    def dijkstra(self, src ):

        dist = [1e7] * self.V

        dist [src] = 0

        processed = [False] * self.V

        for cout in range(self.V):

            # Pick the minimum distance vertex from
            # the set of vertices not yet processed.
            # u is always equal to src in first iteration
            uv = self.minDistance(dist, processed )

            # Put the minimum distance vertex in the
            # shortest path tree
            processed [ uv ] = True

            # Update distance value of the adjacent vertices
            # of the picked vertex only if the current
            # distance is greater than new distance and
            # the vertex in not in the shortest path tree
            for v in range(self.V):
                if (self.graph[ uv ][ v ] > 0 and
                    processed [ uv ] == False and
                    dist [ uv ] > dist [ cout ] + self.graph[ uv ][ v ]):
                    dist [ uv ] = dist [ cout ] + self.graph[ uv ][ v ]

        self.printSolution(dist)
```

源图 / 表 / 代码：英文 PDF 第 35 页 · A8.SS3.fig4

代码注释逐项：Python 实现 Dijkstra 单源最短路，图用邻接矩阵表示；辅助函数从尚未进入最短路树的顶点里寻找距离最小者；初始化下一节点的最小距离；搜索未进入最短路树的最近顶点；实现邻接矩阵图的单源最短路；从尚未处理的顶点中选最小距离顶点，首次迭代 u 总等于 src；将该顶点加入最短路树；仅当现有距离大于新距离、且邻点未进入树时，更新所选顶点的相邻顶点距离。输出表头 Vertex / Distance from Source = 顶点 / 距源点距离。原生成代码中的索引等实现保持原样。

Appendix I Textual generations

附录 I 文本生成例子

S167 · Original / 原文 · PDF p.36 · A9.p1.1

We present below example generations for the proposed method together with several baseline methods. We provide both conditional and unconditional generations. For the conditional generations, we mark the prompt in gray.

下文展示本文方法及若干基线的生成例子，包括条件与无条件生成。条件生成中的提示用灰色标出。译注：以下是模型生成样例，保留了原样例的错误、断裂和事实混淆，不能作为新闻或人物事实使用。

I.1 Conditional generation

I.1 条件生成

S168 · Original / 原文 · PDF p.36 · A9.SS1.fig1.1

The United States on Wednesday asked the UN Security Council to slap an oil embargo on North Korea and freeze the assets of leader Kim Jong-un, in response to Pyongyang's response to the revelations it had restarted its nuclear work in March. "We will continue working to use maximum international pressure on North Korea to agree to the suspension of its nuclear program and reinstate sanctions," said John Bolton, who served as National Security Advisor and Secretary of State under US President Bill Clinton. "Here is North Korea's response to our sanctions," Bolton wrote in a letter to House Minority Leader Nancy Pelosi. "We want you to know that the international community is seriously monitoring North Korea at this time. North Korea is still complying with our requests from the past few days," Bolton said on Monday. "We have been working through the United Nations to provide the information that they gave us." Asked to whether any international pressure will be put in place for North Korea to give up its nuclear weapons, Bolton said the United States can use maximum pressure to get North Korea to abandon its nuclear weapons if it wants. "We've been working to use maximum pressure on North Korea, through the Security Council, and we will continue to do so," said White House Deputy Press Secretary Sarah Huckabee Sanders in Washington. "We're committed to taking any steps necessary to help North Korea pursue its only option for peace, including in this period," she added. The United States did not plan to produce any more oil at this time last year and had not planned to do so this year. "We believe that the North Korea approach is misguided in moving forward with its nuclear program to endanger peace and security in its homeland and neighbors in Asia," said Bolton, adding that the US supplies its nuclear weapons. "We don't want them to sell their nuclear weapons to other nations," he said. Bolton said the US would look for pressure on North Korea, which has been known to use nuclear weapons, as leverage to negotiations with the US. "I will reiterate what I have said before. So, the US has to put pressure on North Korea. But who else is going to hold the cards? Somebody else has to hold the cards," Mr Bolton said. Bolton described what the United States is prepared to do to get North Korea to agree to give up its weapons and asks for sanctions. "As far as I know, we have to use the pressure the reason for putting sanctions on North Korea," he said, adding that the US does not plan to ask the UN Security Council for sanctions alone.

美国周三要求联合国安理会对朝鲜实施石油禁运，并冻结领导人金正恩的资产，以回应平壤对“它在三月重启核工作”这一披露的回应。约翰·博尔顿说：“我们将继续通过最大国际压力，让朝鲜同意暂停核计划并重新实施制裁。”样例称，他曾在美国总统比尔·克林顿任内担任国家安全顾问与国务卿。博尔顿在给众议院少数党领袖南希·佩洛西的信里写道：“这是朝鲜对我们制裁的回应。”他周一说：“我们希望你知道，国际社会此时正在严密关注朝鲜。朝鲜仍在遵守我们过去几天的要求。我们一直通过联合国提供他们给我们的信息。”被问及是否会以国际压力促使朝鲜放弃核武器，他说美国可以使用最大压力，只要朝鲜愿意，就让它放弃核武器。白宫副新闻秘书莎拉·赫卡比·桑德斯在华盛顿说：“我们一直通过安理会向朝鲜施加最大压力，并将继续这么做。”她补充：“包括这段时期在内，我们致力于采取一切必要步骤，帮助朝鲜追求它唯一的和平选择。”美国去年此时没有计划增加石油产量，今年也没有。博尔顿说：“我们认为，朝鲜推进核计划、危及本土及亚洲邻国和平安全的做法是误入歧途”，并补充称美国向它提供核武器。他说：“我们不希望他们把核武器卖给其他国家。”他称美国会对已知使用核武器的朝鲜寻找压力，作为对美谈判的筹码。“我重申之前的话。美国必须施压朝鲜。但还有谁来掌握筹码？必须由另一个人掌握筹码。”博尔顿描述美国准备怎样促使朝鲜同意放弃武器并要求制裁：“据我所知，我们必须利用压力——这就是制裁朝鲜的原因。”他补充，美国并不打算只要求联合国安理会制裁。

S169 · Original / 原文 · PDF p.36 · A9.SS1.fig2.1

The defender is available for the Maribor first leg but his club believed he should be suspended. SNS Group Celtic made an administrative blunder which saw Efe Ambrose left behind in the midfield in the Maribor department and has given him a potential three-match ban today. Although Efe Ambrose will be suspended next Friday, according to reports in Scottish media, the Celtic defender will still be fit for the Champions League first leg at Celtic Stadium in the middle of August. However, the Celtic club wrongly thought that Efe should only receive a three-match ban because he is available for the first leg. Although Efe Ambrose may receive a three-match ban next Friday, Efe Ambrose was part of the Celtic squad for the last match against Liverpool. However, says SNS Group Celtic he was making a tactical error and was left behind in midfield. It is understood that Efe Ambrose did not make the final squad and only played 11 games for the club this season. Efe Ambrose made his professional debut for Celtic in 2008 but spent nine months looking for a new club to return to. With a career-high 72 Celtic appearances, Efe is among Celtic's most capped players ever.

这名后卫可参加对马里博尔的首回合，但其俱乐部认为他应被停赛。SNS Group：凯尔特人犯了行政错误，把埃菲·安布罗斯留在马里博尔部门的中场，并在今天使他面临三场禁赛。苏格兰媒体称，尽管他下周五将被停赛，这名凯尔特人后卫仍可在八月中旬于凯尔特人球场参加欧冠首回合。然而俱乐部错误地认为，既然他可以踢首回合，就只应受到三场禁赛。虽然他可能下周五收到三场禁赛，安布罗斯仍在上次对利物浦的大名单内。但 SNS Group 凯尔特人说，他犯了战术错误，被留在中场。据了解，他未进入最终名单，本赛季只为俱乐部踢了 11 场。样例称，他在 2008 年为凯尔特人完成职业首秀，却用九个月寻找可回归的新俱乐部；以生涯最高的 72 次凯尔特人出场，他属于俱乐部史上出场最多的球员之一。

S170 · Original / 原文 · PDF p.37 · A9.SS1.fig3.1

Carl Jara aka Grain Damaged is an award-winning, professional sand sculptor from Cleveland, Ohio. Jara says he has known since high-school that he wanted to be an artist. After studying English and Art History at the Northeastern University, Jara says one semester he started carving a custom sculpture into sand molds, but didn't know how to do it. With the help of an instructor, he found out and learned how to use rubber molds to make art. Later, he made the decision to learn how to use sand and sculpt himself. In addition to how he makes his own sculptures, Jara says he does special events for comics companies such as Stan Lee and also for institutions like local community colleges and universities. In November of this year, he won the WWHS, The Very Art Of The Very Things Cleveland competition. Afterward, he will continue carving for clients in the comics industry and looks forward to sand sculpting in Cleveland. The Artist is professional sculptor who has been making art, for over 25 years, in various shapes and sizes.The artist says art is all about relationships and the best way to go into the heart is to create art. The artist has taught in various high schools in the Cleveland area and has taught a full time Honors Studio for High School students in grades 6, 7, and 8 time for over 20 years. Since Art is a personal form of artistic expression, he works individually in a way that allows the student that his work engages their imagination and presents ideas in ways that inform and challenge their own paths. Miguel Romano is a professional artist who worked in 3D modeling and animation in the areas of web design and production. The artist currently works as a digital artist in the ad and mass communication industries. In coming to Concrete Cleveland, he is excited to apply the 3D development and production skills he have to his work. The artist has a BFA in sculpture from Ohio University, along with an MFA in sculpture. We look forward to seeing his work very soon! Designed and installed by Concrete This Week. He is a guy originally from Cleveland, Ohio where he pursued a career as a nurse. He then moved to the Atlanta, GA area where he returned to school with a BSN and a BS in nursing and is a licensed nurse. He is a proud sorority brother and still has extra-curricular, as well as taking music lessons and the occasional play. He is a lovely asset at Concrete Cleveland and looks forward to seeing concrete

卡尔·哈拉，别名 Grain Damaged，是来自俄亥俄州克利夫兰的获奖职业沙雕艺术家。他说自己高中起就想做艺术家。在东北大学学习英语和艺术史后，有一个学期开始在沙模中雕刻定制作品，却不知道如何下手；在教师帮助下学会用橡胶模具创作，后来决定学习亲手用沙雕刻。除制作个人雕塑外，他说自己为斯坦·李等漫画公司及当地社区学院、大学等机构举办特殊活动。今年十一月，他赢得 WWHS "克利夫兰万物之艺术" 比赛，此后会继续为漫画行业客户雕刻，并期待在克利夫兰从事沙雕。这位艺术家是专业雕塑家，二十五年多以来制作各种形状、尺寸的艺术品。他说艺术关乎关系，走入内心的最好方式是创作。他曾在克利夫兰多所高中授课，并为六、七、八年级学生教授全日制荣誉工作室二十多年。艺术是个人表达，他以个别方式工作，使学生的想象力受到调动，用能启发、挑战他们自身道路的方式呈现观念。米格尔·罗马诺是一名专业艺术家，曾在网页设计与制作领域从事三维建模、动画，现于广告及大众传播行业任数字艺术家。来到 Concrete Cleveland，他期待把三维开发制作技能应用于工作。他拥有俄亥俄大学雕塑 BFA 及雕塑 MFA。我们期待很快看到他的作品！由 Concrete This Week 设计并安

装。他原籍俄亥俄州克利夫兰，曾从事护理；后迁至佐治亚州亚特兰大，返校获得护理 BSN 与 BS 学位，并取得护士执照。他为自己是联谊会兄弟感到自豪，仍有课外活动、音乐课与偶尔的戏剧。他是 Concrete Cleveland 的可贵成员，期待看到混凝土——（原样例至此中断）。

I.2 Unconditional generation

I.2 无条件生成

S171 · Original / 原文 · PDF p.38 · A9.SS2.fig1.1

Here’s how that strategy works for your job: 1) You now plan upon what you accomplish to fulfill your goals. Management cannot plan what happens to you. This may not be your ultimate personal decision, but it’s perfectly fine to look at it. You just need to make sure you want to achieve this. Now, because you’ve worked at goal, you don’t have to talk about your status tomorrow, after all, you have to do your job and take care of yourself. Next steps, there may be some work to do. There is a company down the road you right – literally millions of things that would have to be done. But of course it would have a different outlook. If you’re going to do something, the customer might not be able to tell you. 2) Between those two steps are the plan in step so that your actions will be executed. Then you have taken other steps (usually a few less important changes), like delivery. If you already know what that means, and you’re having to stay up and take action you can make sure you don’t have to point out in the moment to plan the action. With business goals, it is not easy to pick up what appears best for us. We have to see what really is. What we do. We can’t make a plan on the floor and come back up with exactly what you’re doing. If you want to work every step, then you need to differentiate from the action and what the next step represents. Eventually, you’ll be less motivated to focus on this step and the previous one. Unfortunately if you don’t change your main thing, you may be able to lose your motivation to work on “pivot.” Unless that’s possible, and if you don’t change something, then the task may not be at the right time. Instead of doing something, it is just in advance of your ultimate goal. Although you might make a mistake with every single day to day plan, it still is a great opportunity to correct your mistake, become new and commit to working extra hours and meeting your goals promptly. The truth is, everything goes right for you no matter how quick a decision you become. The customer will never allow you to make the worst decisions. Otherwise, you make the very first decision. 3) Take timing as part of action. If you don’t feel like you can keep it, a plan without help of timing stops you from doing. When it’s like your plan in action can lead to something such as this: Now that you know what to do. For example, you might live in a place in the building that serves every customer, has 3 employees per team, and 3 clients on one. You will get things done the next day. Change your performance is the first step towards greater success, for example. Your team, at this point in the Customer department, will know how the customer deal with a single employee, the level chain, and more. Make sure you take action now that you change it. As a company, it won’t be hard but you will have lots of work to do when you change. 4) Make sure it’s your night.

以下是这种策略如何用于你的工作：1）现在你会规划完成什么来实现目标。管理层不能规划发生在你身上的事。这也许不是最终个人决定，但看看它完全没问题；只需确认你想实现它。现在，因为你已经朝目标工作，明天便不必谈自己的状态，毕竟你必须工作、照顾自己。下一步也许还有事要做。前方有一家你右边的公司——确实有数百万件事要做。当然它会有不同看法。如果你要做某事，客户可能无法告诉你。2）这两步之间有计划中的步骤，使行动得以执行。然后你又采取其他步骤，通常是少量较不重要的变化，例如交付。如果已经知道这意味着什么，而你必须熬夜行动，就能确保不必在那个时刻指出要规划行动。有商业目标时，很难拿起看上去最适合我们的东西。必须看清实际是什么、我们做什么。不能在地板上订计划，再回上来时恰好知道你在做什么。若想每一步都工作，就必须区分行动以及下一步代表什么。最终，你会较少有动力关注这一步和上一步。遗憾的是，若不改变主要事情，可能会失去去做“转向”的动力。除非那是可能的，而你没有改变什么，任务也许就不在合适的时机。不是做某事，而只是提前于最终目标。尽管每天的计划都可能出错，它仍是纠错、焕新、承诺加班并及时达标的好机会。事实是，不管你多快变成一个决定，一切都会顺利。客户绝不容许你做最差决定；否则你就做第一个决定。3）把时机当作行动的一部分。若觉得无法坚持，缺乏时机帮助的计划会阻止行动。当你的计划付诸行动可能产生这样的结果：现在你知道该做什么。例如，你可能住在一栋为每位客户服务的建筑里，每队有三名员工、每一组有三位客户。你会在第二天完成工作。例如改变表现，是迈向更大成功的第一步。客户部门的团队会知道客户如何与单个员工打交道、等级链条等。确保现在就采取行动去改变。作为公司，这并不难，但改变时会有很多工作。4）确保这是属于你的夜晚。

S172 · Original / 原文 · PDF p.38 · A9.SS2.fig2.1

Watch the humour but also the humanity behind the work we're doing. The truth is something very tragic and delicate in the middle of a very fractured world. It's the one thing that makes me proud. I feel like a singular individual has had to come together with this story. There's a lot of people who I've worked with for the very beginning, because I have got people, you've got people who have just had these eyes on this story, and this sense of what we are, that run through our final movie. It's the very beginning we're at. The very beginning, we're not there, we will get there but we won't need. This is a story and these amazing actors, these fantastic violence, violence, that was just are elements and a complex world of conflict. When something like that is set to build this narrative and you're directing the world of these characters based around their individual needs it's very, extremely confusing, very heartbreaking — it's really quite intense—this was all built within it, and what it is, it's a 35-year old period that was slavery and still was very strong, these guys were operating to the edge and going to the point where we ended up setting up a big narrative, OK, that's good, it's okay, in some ways heroism is a noble imperative that we are fighting against, and recognize maturity as the mercurial nature and these are all human and we've got to clean it up so that that stuff is there and we've got to restore it. And the project we're looking at here is our common goal is that anything can be done to make that happen and everybody can do whatever their want to do and do it as they please. That's the spirit of it. That's the movie I've made with Steven Wright in writing.

看看我们工作的幽默，也看看其中的人性。事实是在一个非常破碎的世界里，有某种十分悲惨而微妙的东西。这是唯一让我自豪的事。我觉得一个独特个体不得不与这个故事走到一起。很多人从最初便与我共事，因为我有这些人，你也有这些人，他们一直以这样的目光看着故事，而“我们是什么”的这种感受贯穿最终电影。我们处在最初，最初；还没到那里，会到，但我们不需要。这是一个故事，这些惊人的演员、这些奇妙的暴力——暴力——只是元素，是复杂的冲突世界。当这些东西被设置来构建叙事，而你又根据人物的个人需求去导演他们的世界时，会非常、极其困惑，非常令人心碎——真的很强烈——这一切都建在其中；它是什么呢，是一段三十五年的时期，那是奴役，仍很强烈；这些人行至边缘，走到我们最后搭建宏大叙事的地方。好，那很好，也没关系。从某些方面说，英雄主义是我们对抗的高尚命令，要把成熟理解为易变本性；这些都是人的，我们必须整理好，使那些东西存在，必须恢复它。我们在这里看的项目，共同目标是任何事都能为使其发生而做，每个人都可以做想做的事、随意去做。精神就在这里。这就是我与史蒂文·赖特在写作中制作的电影。