Lecture 05

Discrete diffusion models and discrete flow matching

MIT IAP 2026 | Jan 30, 2025

Peter Holderrieth and Ron Shprints

Sponsor: Tommi Jaakkola

Class Overview

- **Lecture 1 - Flow and Diffusion Models**
- **Lecture 2 - Flow Matching:** Training algorithm.
- **Lecture 3 - Score Matching, Guidance:** How to condition on a prompt.
- **Lecture 4 - Build Image Generators: Latent spaces + Network architectures**
- **Lecture 5 - Discrete diffusion models and flow matching**

Discrete Diffusion Models in the news

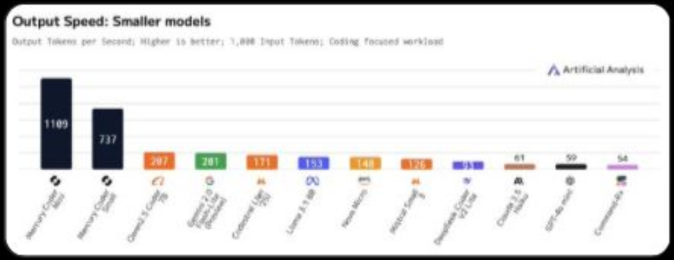
Subscribe To Newsletters **Forbes**

INNOVATION > AI

Generative AI Gets Shaken Up By Newly Announced Text-Producing Diffusion LLMs

Inception Labs @InceptionAILabs · Feb 26

We achieve over 1000 tokens/second on NVIDIA H100s. Blazing fast generations without specialized chips!



SEARCH **FORTUNE**

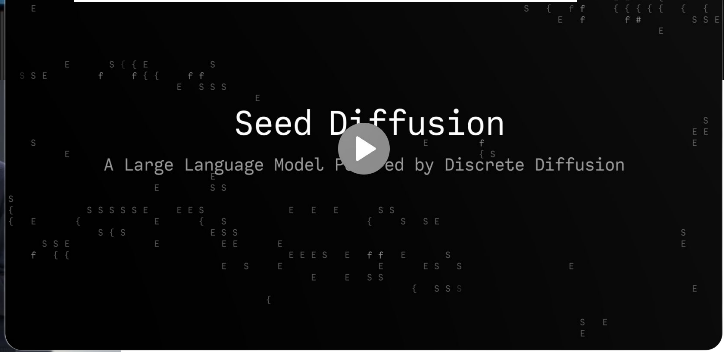
HOME NEWS FORTUNE 500 TECH FINANCE LEADERSHIP LIFESTYLE MULTIMEDIA

TECH · GOOGLE

Gemini Diffusion was the sleeper hit of Google I/O and some say its blazing speed could reshape the AI model wars

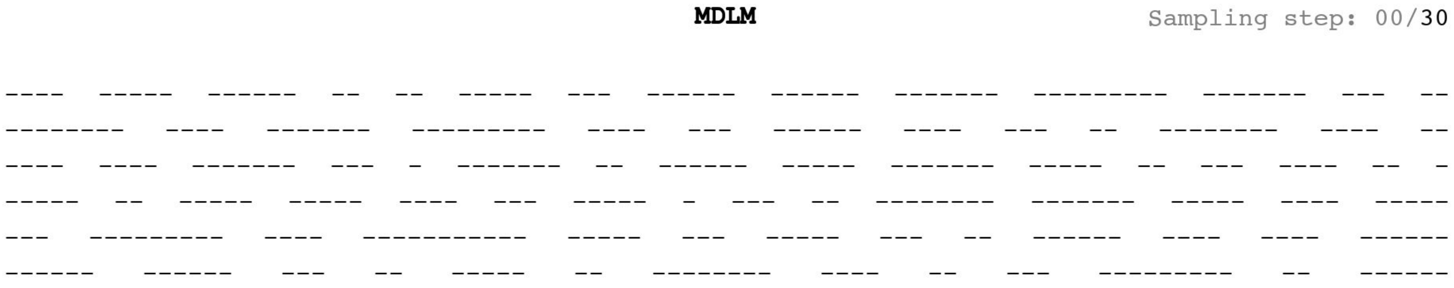
BY SHARON GOLDMAN
AI REPORTER
May 21, 2025 at 5:54 PM EDT

ByteDance | Seed



Slide: Subham Sahoo

Diffusion LLMs generate text in arbitrary order



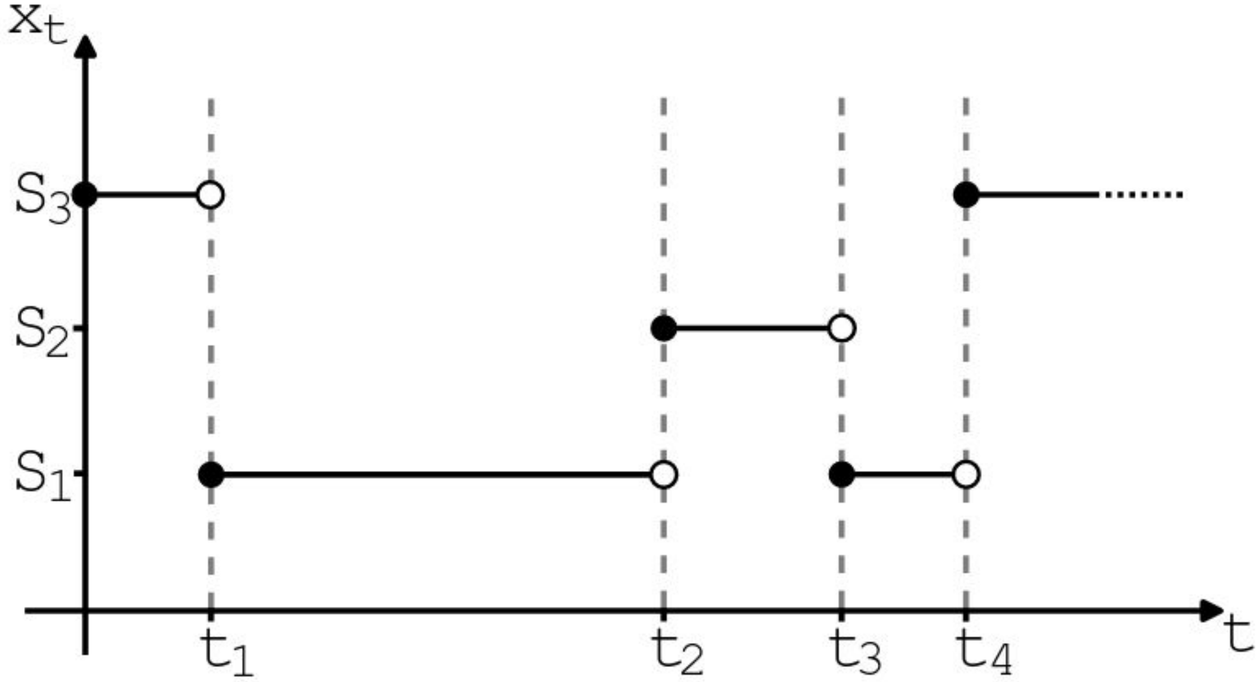
Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces”, NeurIPS 2022

Slide: Subham Sahoo

Discrete diffusion models and discrete flow matching

- Models for **discrete sequence data:** Language, protein sequences, etc.
- *Note: There is no diffusion/SDE and also no flow/ODE in discrete space.*
- Rather: Learning principles of flow matching and denoising can be generalized to discrete data!
- Mathematical model: **Continuous-time Markov chains (CTMCs)**
- Today:
    - CTMC Models
    - Discrete Flow Matching:
        - Discrete Probability Paths
        - Discrete Marginalization Trick
        - Discrete FM objective

Continuous Time Markov Chains



Figures: Andrew Campbell

Example - Continuous-time Markov chain

State space

$$S = \{a, b\}$$

Rate matrix

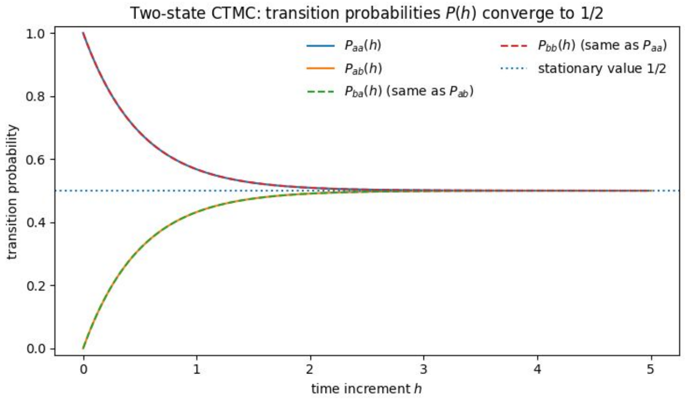
$$Q = \begin{array}{c|cc} & a & b \\ \hline a & -\lambda & \lambda \\ b & \lambda & -\lambda \end{array}$$

By showing evolution equation (taking derivatives), one obtains:

$$\begin{pmatrix} p(X_{t+h} = a | X_t = a) & p(X_{t+h} = a | X_t = b) \\ p(X_{t+h} = b | X_t = a) & p(X_{t+h} = b | X_t = b) \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 + e^{-2\lambda h} & 1 - e^{-2\lambda h} \\ 1 - e^{-2\lambda h} & 1 + e^{-2\lambda h} \end{pmatrix}$$

Convergence for h to infinity:

$$\rightarrow \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$



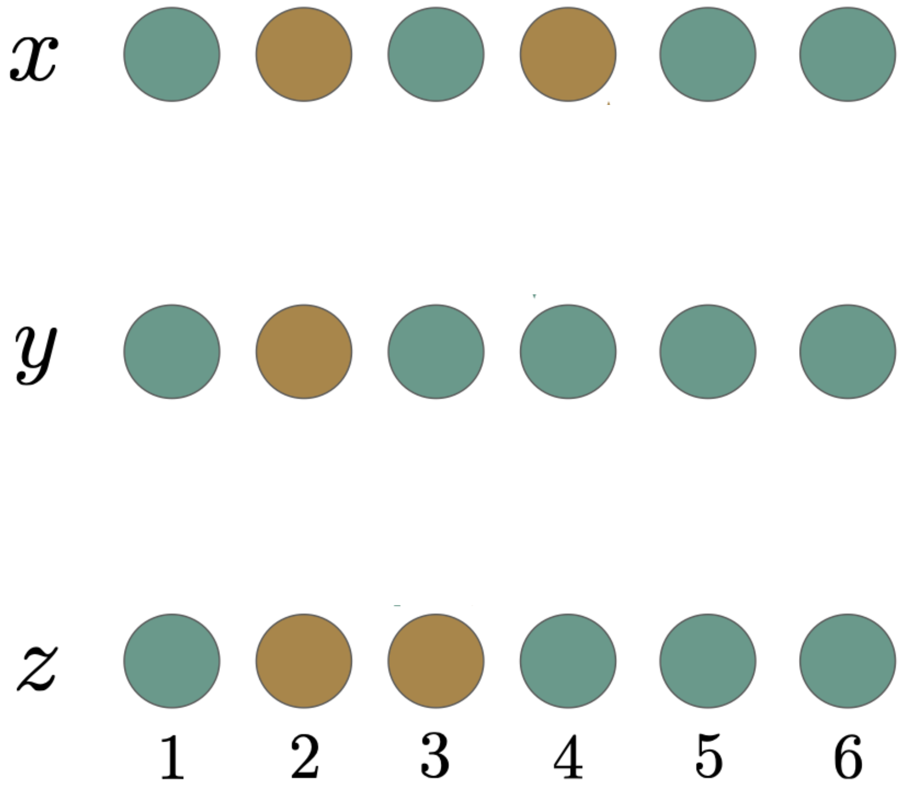
Examples of neighbors

$$Q_t^{\theta}(z|x) = 0$$

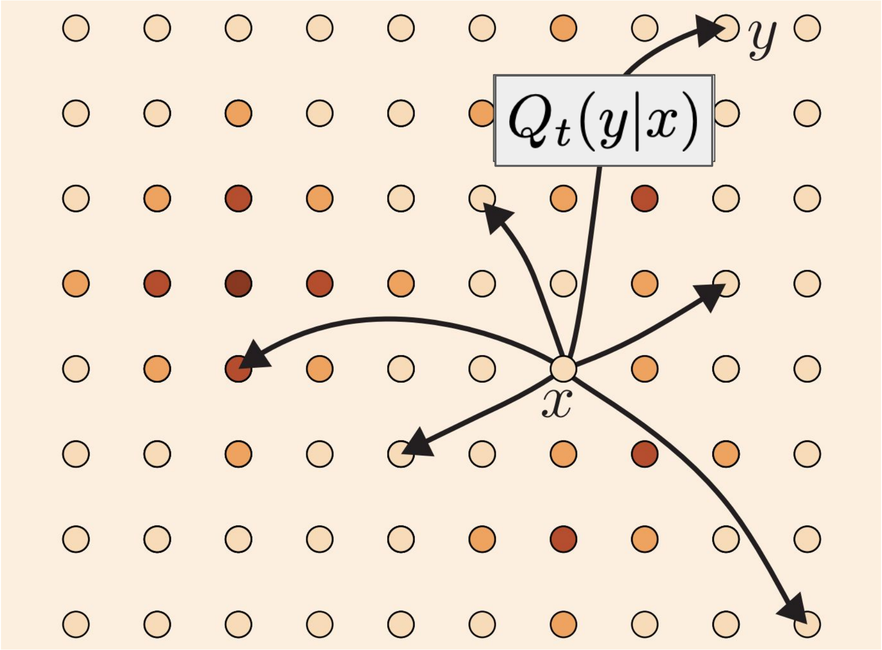
X and y are neighbors

Y and z are neighbors

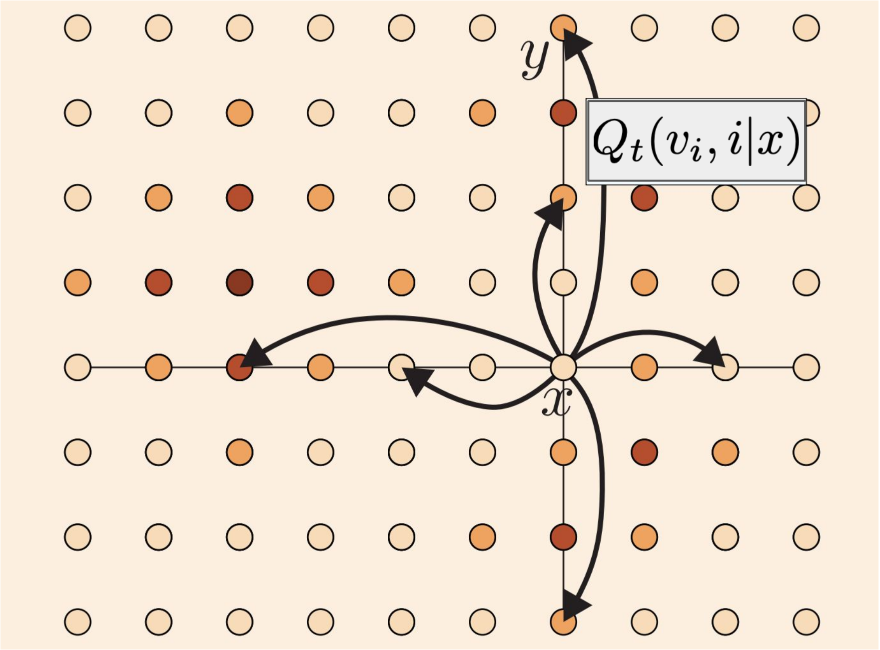
Z and x are not neighbors



General CTMC



Factorized CTMC



Figures: Yaron Lipman

Sampling from factorized CTMC models

Algorithm 7 Sampling from a Factorized CTMC Model (Euler / τ-leaping)

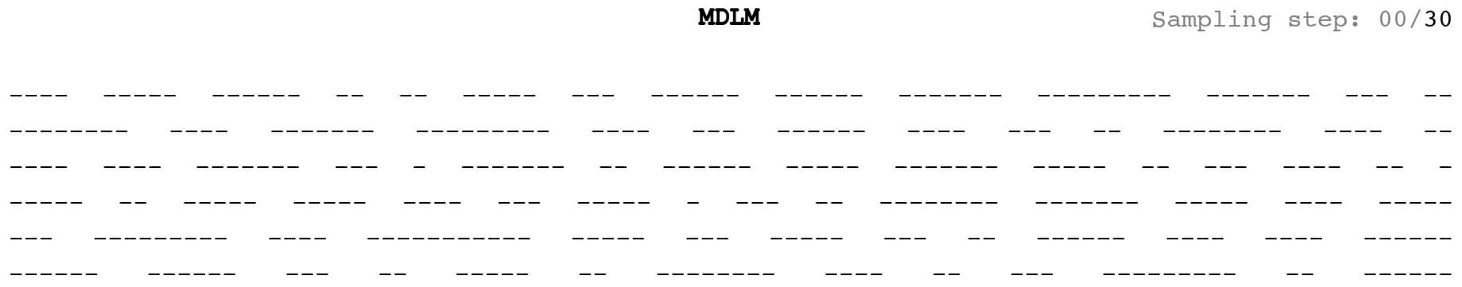
Require: Rate network Q_t^θ (factorized), initial distribution p_{init}, number of steps n

Set $t \leftarrow 0$
Set step size $h \leftarrow \frac{1}{n}$
Draw a sample $X_0 \sim p_{\text{init}}$, where $X_0 = (X_0^{(1)}, \ldots, X_0^{(d)}) \in \mathcal{V}^d$
for $i = 1, \ldots, n$ **do**
    Compute factorized jump rates $\{q_j(v)\}_{j=1..d,\ v\in\mathcal{V}} \leftarrow Q_t^\theta(\cdot \mid X_t)$
    for $j = 1, \ldots, d$ **(in parallel) do**
        $x \leftarrow X_t^{(j)}$ {current token at position j}
        Define the per-position Euler transition probabilities $\tilde{p}_{j,t}(\cdot \mid X_t^{(j)} = x)$ by

$$\tilde{p}_{j,t}(v \mid x) \ = \begin{cases} h\, q_j(v), & v \neq x, \\ 1 - h \displaystyle\sum_{v' \in \mathcal{V}\setminus\{x\}} q_j(v'), & v = x. \end{cases}$$

        Sample $X_{t+h}^{(j)} \sim \textsc{Categorical}(\{\tilde{p}_{j,t}(v \mid x)\}_{v\in\mathcal{V}})$
    end for
    Set $t \leftarrow t + h$
end for
return X_1

Illustration of sampling procedure



Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces”, NeurIPS 2022

Slide: Subham Sahoo

Generative Modeling with CTMCs

Data distribution:

$$p_{\text{data}}(z) \quad (z \in S)$$

E.g. distribution of texts on the internet

Initial distribution:

$$p_{\text{init}}(z) \quad (z \in S)$$

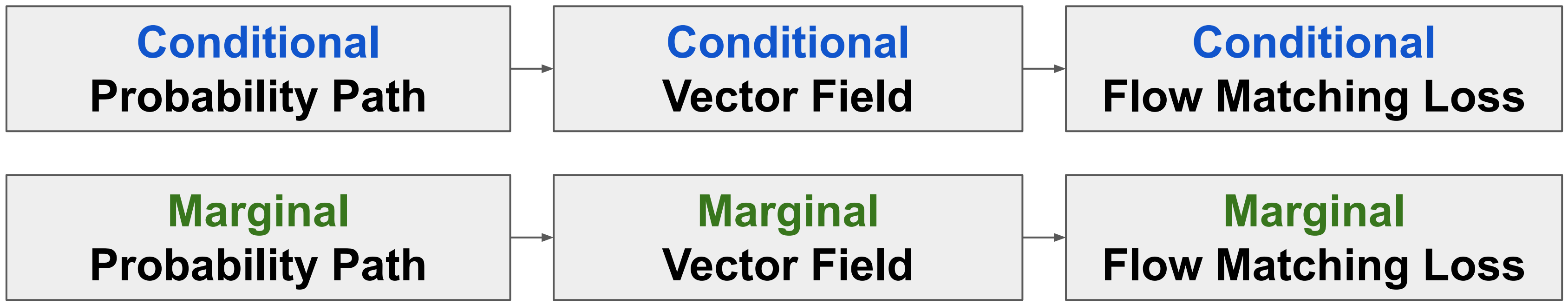
E.g. uniform distribution

$$p_{\text{init}}(z) = \frac{1}{|S|}$$

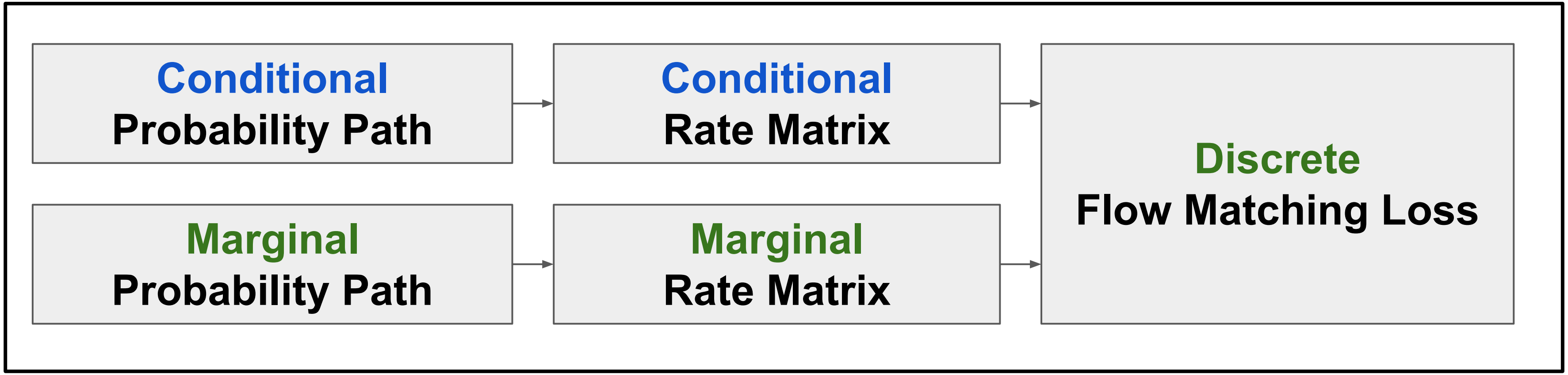
Goal: Convert "Noise" to Data with a CTMC

$$X_0 \sim p_{\text{init}} \xrightarrow{\textit{CTMC}} X_1 \sim p_{\text{data}}$$

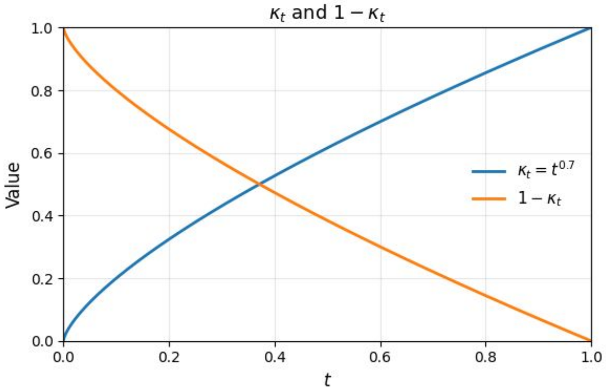
Continuous Flow Matching



The Discrete Flow Matching Matrix



Example - Factorized Mixture Path



Scheduler: $0 \leq \kappa_t \leq 1$ such that

$$\kappa_0 = 0, \kappa_1 = 1$$

Idea: Noise each token independently with probability κ_t

$$p_t(x \mid z) = \prod_{j=1}^{d} \left[(1-\kappa_t)\, p_{\text{init}}^{(j)}(x_j) + \kappa_t\, \delta_{z_j}(x_j) \right]$$

Downweight noise *Upweight data*

Sampling procedure:

$$m_j \sim \text{Bernoulli}(\kappa_t), \quad \xi_j \sim p_{\text{init}}^{(j)}$$

$$x_j = m_j\, z_j + (1-m_j)\, \xi_j, \qquad j = 1, \ldots, d$$

$$x = (x_1, \cdots, x_d)$$

Check for yourself that this is a cond. prob. path!

Toy example

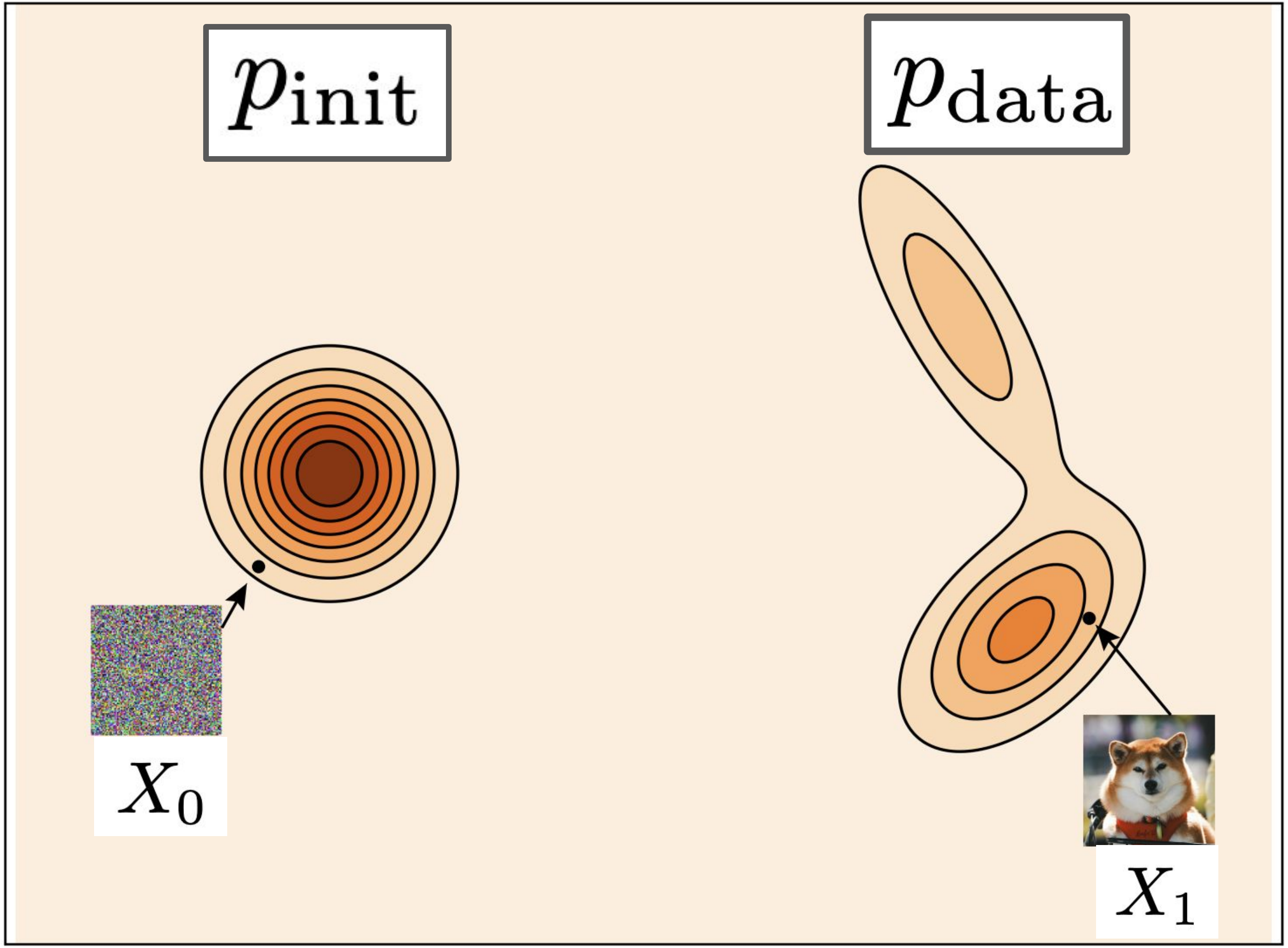


Figure credit: Yaron Lipman

Discrete Probability Path

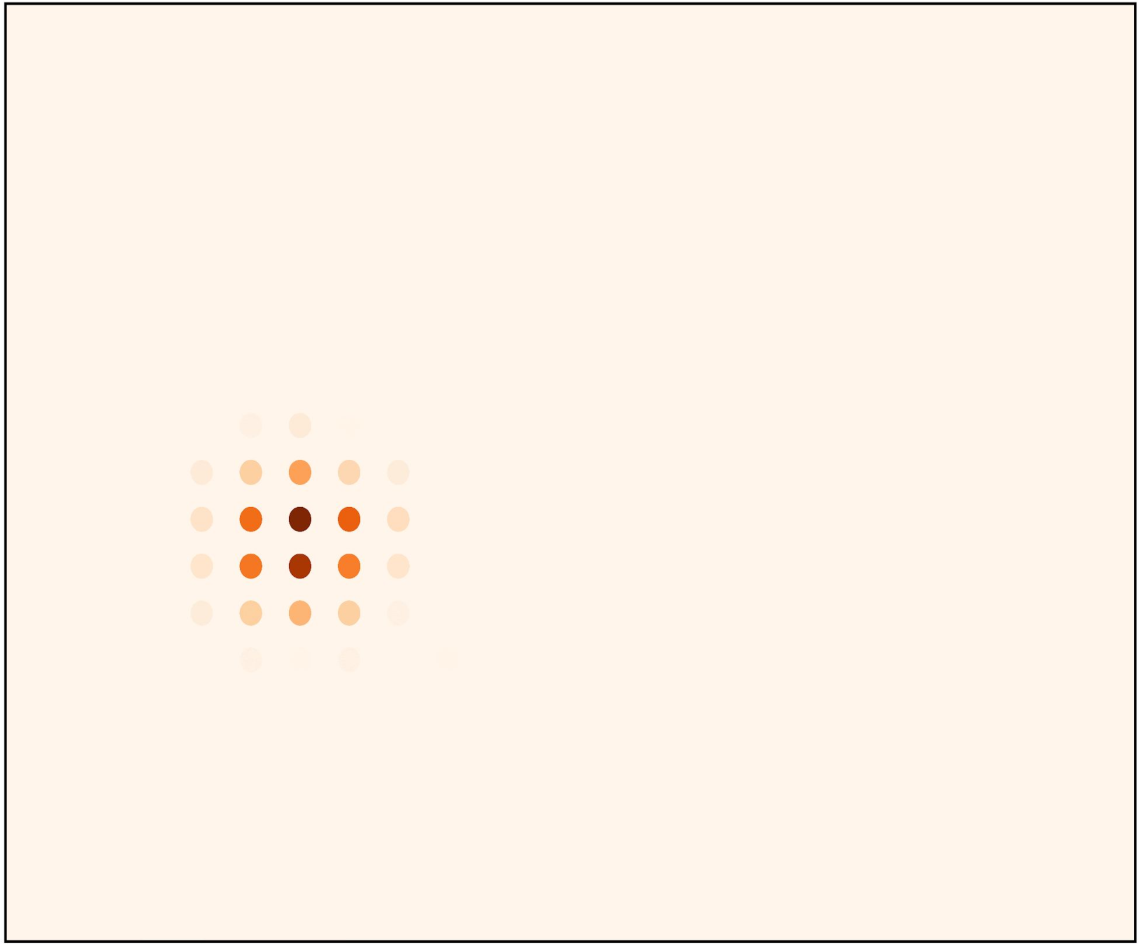


Figure credit: Yaron Lipman

Note difference to before: Probability is teleported - not moved through space

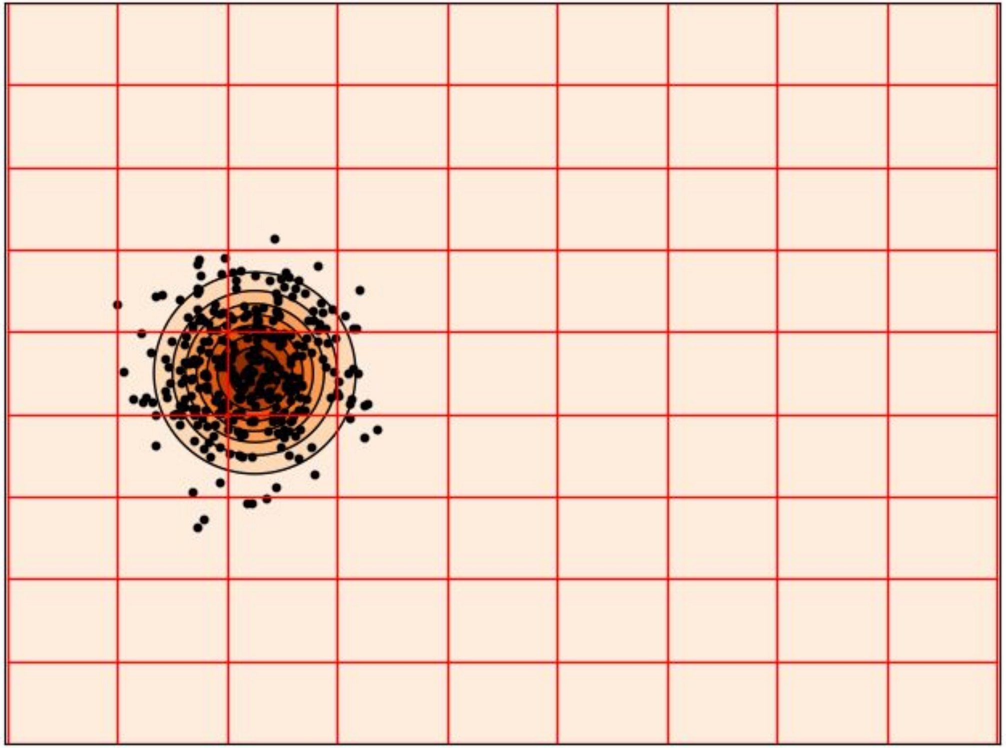


Figure credit: Yaron Lipman

Discrete Probability Path

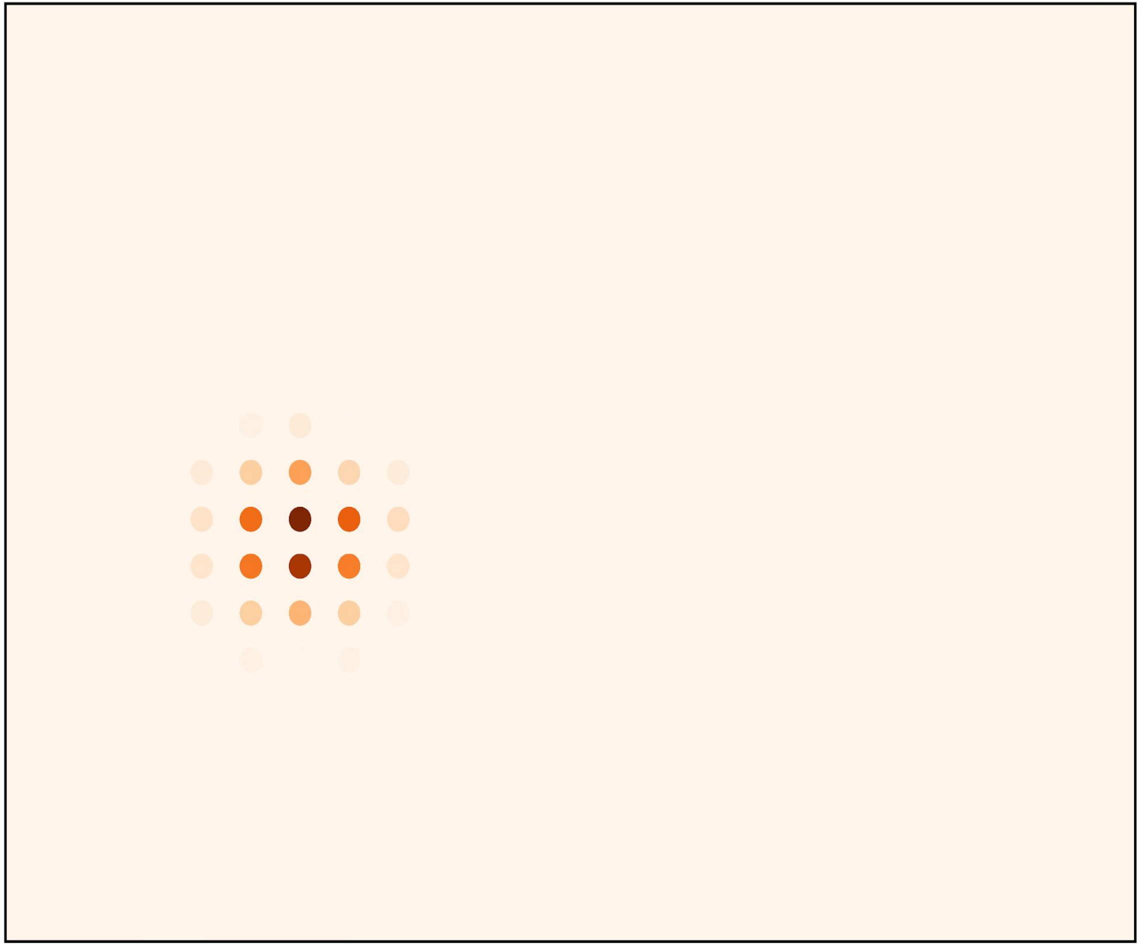


Figure credit: Yaron Lipman

Kolmogorov Forward Equation

Discrete analogue to the continuity equation.

A CTMC with rate matrix Q_t follows the probability path

$$X_t \sim p_t \quad (0 \leq t \leq 1)$$

if and only if the Kolmogorov Forward Equation (KFE) holds:

$$\frac{\mathrm{d}}{\mathrm{d}t} p_t(x) = \sum_{y \in S} Q_t(x|y) p_t(y)$$

change of probability

Net inflow

Conditional Rate Matrix for Factorized Mixture Path

The conditional rate matrix for factorized mixture path is factorized (i.e. rates only non-negative for one token updates) and given by

$$Q_t^z(y|x) = (Q_t^z(v_i, j|x_j))_{v_i, j}$$

$$Q_t^z(v_i, j|x_j) = \frac{\dot{\kappa}_t}{1-\kappa_t}(\delta_{z_j}(v_i) - \delta_{x_j}(v_i))$$

$$= \frac{\dot{\kappa}_t}{1-\kappa_t}\begin{cases} 0 & \text{if } x_j = z_j \\ 1 & \text{if } v_i = z_j, x_j \neq z_j \\ 0 & \text{if } v_i \neq z_j, x_j \neq z_j \\ -1 & \text{if } v_i = x_j, x_j \neq z_j \end{cases}$$

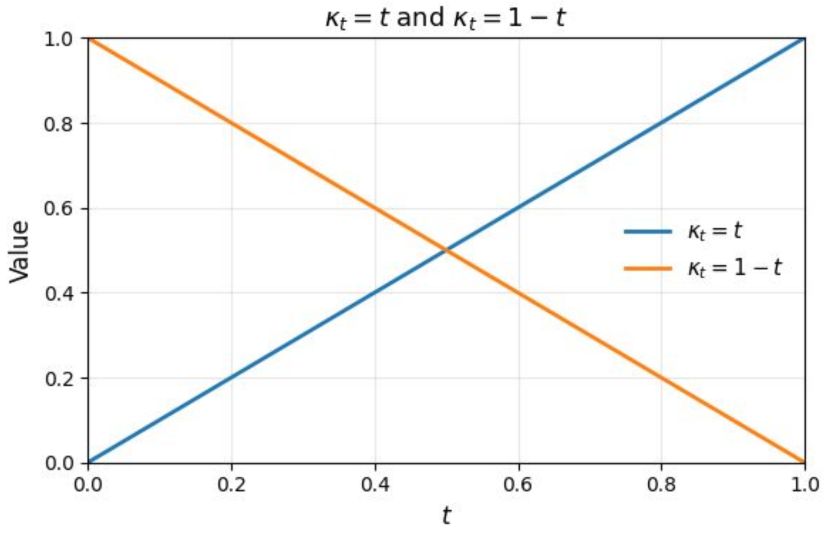
If current token correct, zero rate

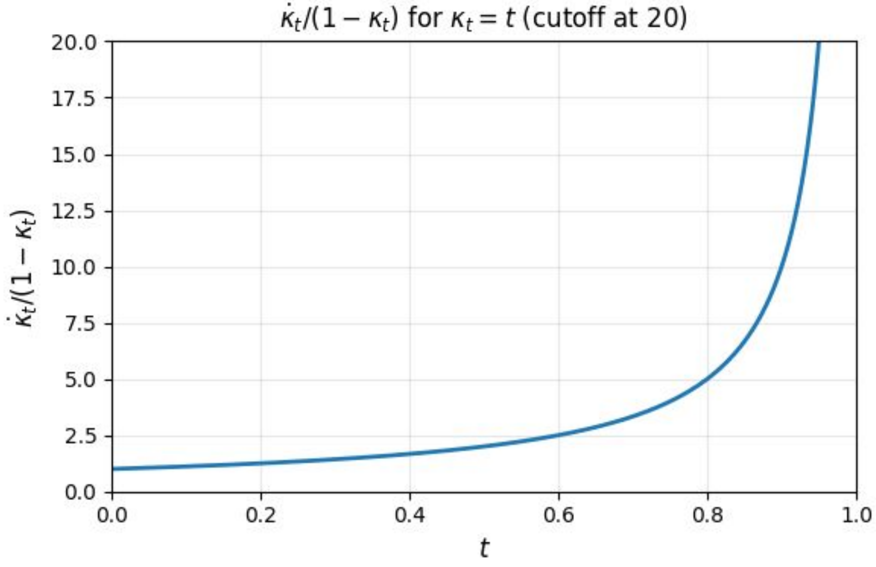
If incorrect, jump to correct token

Outgoing rate from current token, if incorrect

$$\kappa_t$$

$$\frac{\dot{\kappa}_t}{1-\kappa_t}$$





Conditional Rate Matrix for Factorized Mixture Path

$$Q_t^z(y|x) = (Q_t^z(v_i, j|x_j))_{v_i, j}$$

$$Q_t^z(v_i, j|x_j) = \frac{\dot{\kappa}_t}{1-\kappa_t}(\delta_{z_j}(v_i) - \delta_{x_j}(v_i))$$

$$= \frac{\dot{\kappa}_t}{1-\kappa_t}\begin{cases} 0 & \text{if } x_j = z_j \\ 1 & \text{if } v_i = z_j, x_j \neq z_j \\ 0 & \text{if } v_i \neq z_j, x_j \neq z_j \\ -1 & \text{if } v_i = x_j, x_j \neq z_j \end{cases}$$

Rates explode at t=1

If current token correct, zero rate

If incorrect, jump to correct token

Outgoing rate from current token, if incorrect

Conditional Prob. Path, Vector Field, and Score

	Notation	Key property	Factorized mixture
Conditional Probability Path	$p_t(x\|z)$	Interpolates p_{init} and a data point z	$\prod_{j=1}^{d}\left[(1-\kappa_t)\, p_{\text{init}}^{(j)}(x_j) + \kappa_t\, \delta_{z_j}(x_j)\right]$
Conditional Rate Matrix	$Q_t^z(y\|x)$	CTMC follows conditional path	$Q_t^z(y\|x) = (Q_t^z(v_i, j\|x_j))_{v_i,j}$ $Q_t^z(v_i, j\|x_j) = \frac{\dot{\kappa}_t}{1-\kappa_t}(\delta_{z_j}(v_i) - \delta_{x_j}(v_i))$

Marginal Prob. Path, Vector Field, and Score

	Notation	Key property	Formula
Marginal Probability Path	p_t	Interpolates p_{init} and p_{data}	$\sum_{z \in S} p_t(x\|z) p_{\text{data}}(z)$
Marginal Vector Field	$Q_t(y\|x)$	CTMC follows marginal path	$\sum_{z \in S} Q_t^z(y\|x) \frac{p_t(x\|z) p_{\text{data}}(z)}{p_t(x)}$

Marginal Rate Matrix for Factorized Mixture Path

Conditional Rate Matrix

$$Q_t^z(y|x) = (Q_t^z(v_i, j|x_j))_{v_i,j}$$

$$Q_t^z(v_i, j|x_j) = \frac{\dot{\kappa}_t}{1-\kappa_t}(\delta_{z_j}(v_i) - \delta_{x_j}(v_i))$$

Known terminal point!

Marginal Rate Matrix

$$Q_t(y|x) = (Q_t(v_i, j|x))_{v_i,j}$$

$$Q_t(v_i, j|x) = \frac{\dot{\kappa}_t}{1-\kappa_t}(p_{1|t}(z_j = v_i|x) - \delta_{x_j}(v_i))$$

Conditional probability!

Only unknown!

Discrete Flow Matching loss

Posterior probability network

$$p^{\theta}_{1|t}(z_j|x)$$

Learn via classification:

$$\mathcal{L}_{\mathrm{DFM}}(\theta) = \mathbb{E}_{z\sim p_{\mathrm{data}}, t\sim \mathrm{Unif}_{[0,1]}, x\sim p_t(\cdot|z)} \left[\sum_{j=1}^{d} -\log p^{\theta}_{1|t}(z_j|x) \right]$$

Cross-entropy loss for every dimension!

Algorithm 8 Training factorized CTMC Model (Discrete Diffusion)

Require: Dataset of sequences $z \sim p_{\text{data}}$ with $z = (z_1, \dots, z_d) \in \mathcal{V}^d$;
initial (noise) token marginals $p_{\text{init}}^{(j)}$ on $\mathcal{V}$; schedule $\kappa_t \in [0, 1]$;
posterior network f_θ returning per-position logits over $\mathcal{V}$; optimizer $\textsc{Opt}$

for each training iteration **do**
  Sample a data point $z \sim p_{\text{data}}$
  Sample time $t \sim \text{Unif}[0, 1]$ and compute $\kappa \leftarrow \kappa_t$
  Sample a noisy state $x \sim p_t(\cdot \mid z)$ (factorized mixture path):
  for $j = 1, \dots, d$ **(in parallel) do**
    Sample mask $m_j \sim \text{Bernoulli}(\kappa)$
    Sample noise token $\xi_j \sim p_{\text{init}}^{(j)}$
    Set $x_j \leftarrow m_j\, z_j + (1 - m_j)\, \xi_j$
  end for
  $x \leftarrow (x_1, \dots, x_d)$
  Predict terminal-token posteriors via logits from the network:

$$\ell_j(\cdot) \;\leftarrow\; f_\theta(x, t)_j \qquad \Rightarrow \qquad p^\theta_{1|t}(v \mid x)_j \;=\; \text{Softmax}\big(\ell_j\big)(v)$$

  Discrete Flow Matching loss (token-wise NLL of z):

$$\mathcal{L}_{\text{DFM}}(\theta) \;\leftarrow\; \sum_{j=1}^{d} \Big[-\log p^\theta_{1|t}(z_j \mid x)_j \Big]$$

  Update parameters: $\theta \leftarrow \textsc{Opt.step}\big(\nabla_\theta \mathcal{L}_{\text{DFM}}(\theta)\big)$
end for

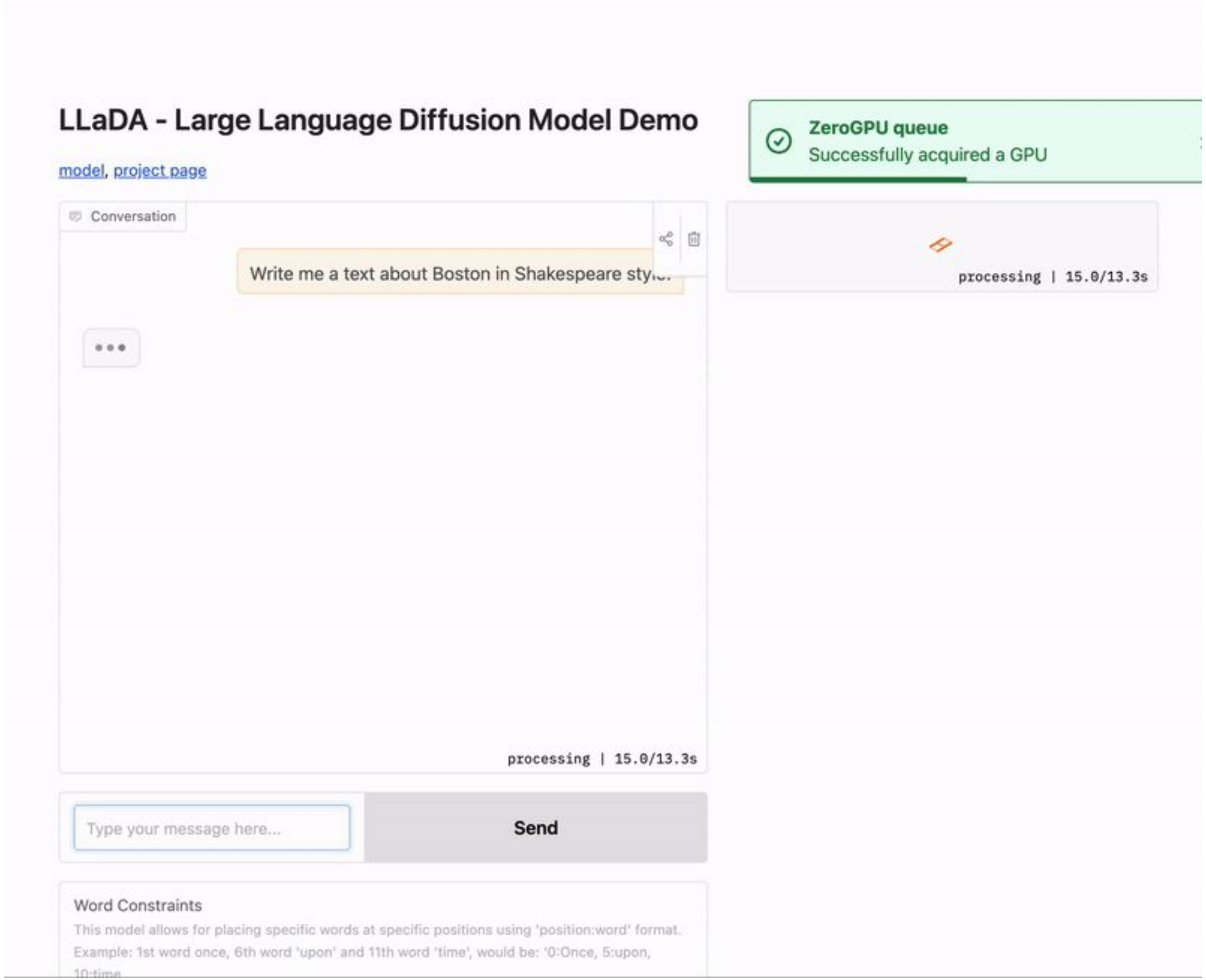
Mask Diffusion Language Models

Introduce new token into vocabulary: **[MASK]**

This token indicates that we masked the reference token

Initial distribution: $$\delta_{[\mathrm{MASK}]}$$

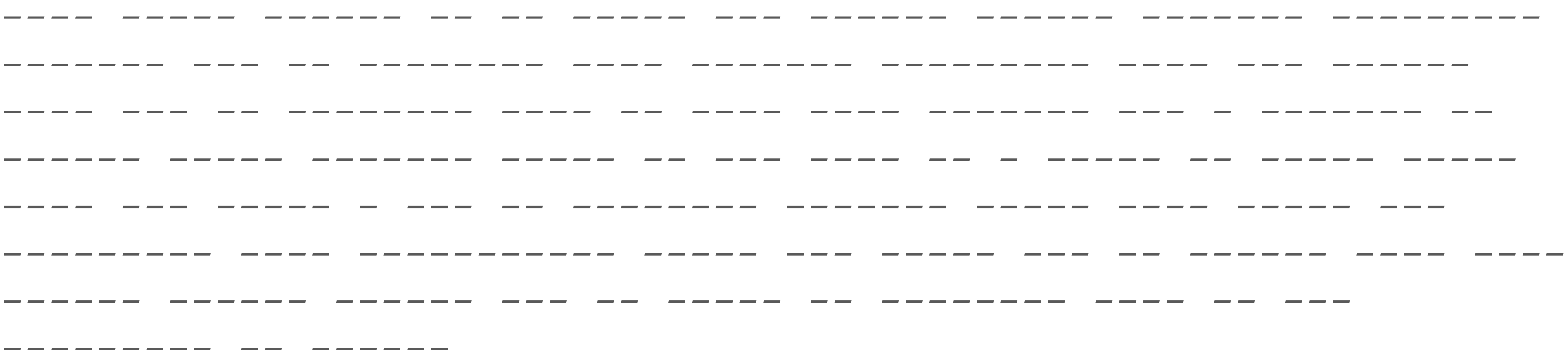
____ _____ ______ __ __ _____ ___ ______ ______ _______ _________
_______ ___ __ ________ ____ _______ _________ ____ ___ ______
____ ___ __ ________ ____ __ ____ ____ _______ ___ _ _______ __
______ _____ _______ _____ __ ___ ____ __ _ _____ __ _____ _____
____ ___ _____ _ ___ __ ________ _______ _____ ____ _____ ___
_________ ____ ___________ _____ ___ _____ ___ __ ______ ____ ____
______ ______ ______ ___ __ _____ __ ________ ____ __ ___
_________ __ ______


LLaDA - Large Language Diffusion Model Demo
model, project page
ZeroGPU queue
Successfully acquired a GPU
Conversation
Write me a text about Boston in Shakespeare style.
processing | 15.0/13.3s
processing | 15.0/13.3s
Type your message here...
Send
Word Constraints
This model allows for placing specific words at specific positions using 'position:word' format. Example: 1st word once, 6th word 'upon' and 11th word 'time', would be: '0:Once, 5:upon, 10:time

Slide: Subham Sahoo

Sampling from a Masked Language Model

Start with fully masked (initial distribution)



Slide: Subham Sahoo

t=0.3

---- ----- ------ -- -- ----- --- ------ squad, ------- ---------
------- --- to remember ---- ------- --------- when --- ------
---- --- to -------- ---- -- ---- ---- ------- --- - ------- --
twenty adobe ------- ----- -- --- ---- -- - ----- of ----- water
---- --- along - --- -- polished ------- which ---- ----- ---
--------- ---- prehistoric ----- --- ----- --- -- ------ ---- many
------ ------ ------ --- in ----- -- -------- ---- -- ---
--------- -- ------

Slide: Subham Sahoo

t=0.6

Many years later, as he faced --- ------ squad, ------- --------- Buendía was to remember that distant --------- when his ------ took --- to discover ---- At that ---- ------- was a village of twenty adobe houses, built -- the bank of - river of clear water that ran along - --- of polished stones, which were ----- and --------- like prehistoric eggs. --- ----- --- so recent ---- many things lacked names, and in ----- -- -------- them it was --------- to ------

t=0.8

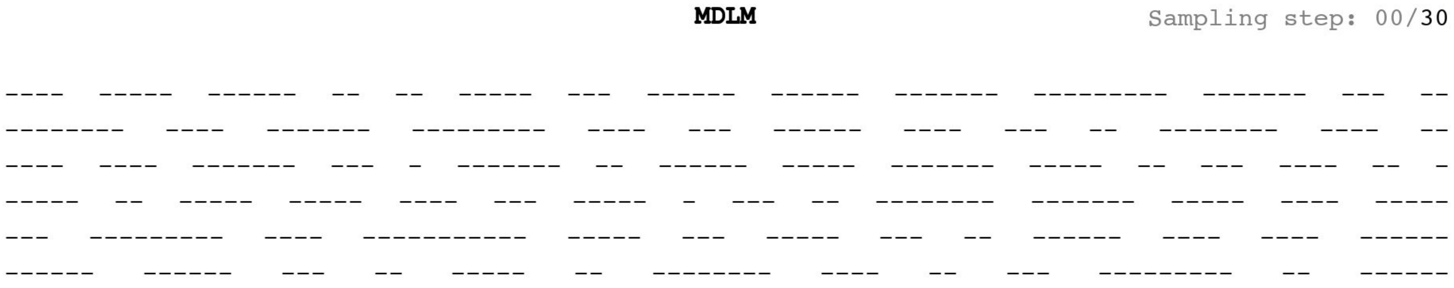
Many years later, as he faced the firing squad, Colonel Aureliano Buendía was to remember that distant --------- when his father took him to discover ice. At that time Macondo was a village of twenty adobe houses, built on the bank of - river of clear water that ran along a bed of polished stones, which were white and enormous, like prehistoric eggs. The world --- so recent that many things lacked names, and in order -- indicate them it was --------- to point.

Slide: Subham Sahoo

t=1.0

Many years later, as he faced the firing squad, Colonel Aureliano Buendía was to remember that distant afternoon when his father took him to discover ice. At that time Macondo was a village of twenty adobe houses, built on the bank of a river of clear water that ran along a bed of polished stones, which were white and enormous, like prehistoric eggs. The world was so recent that many things lacked names, and in order to indicate them it was necessary to point.

Masked Diffusion LLMs generate text



Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces”, NeurIPS 2022

Slide: Subham Sahoo

Discussion: Discrete Diffusion Models vs Autoregressive Models

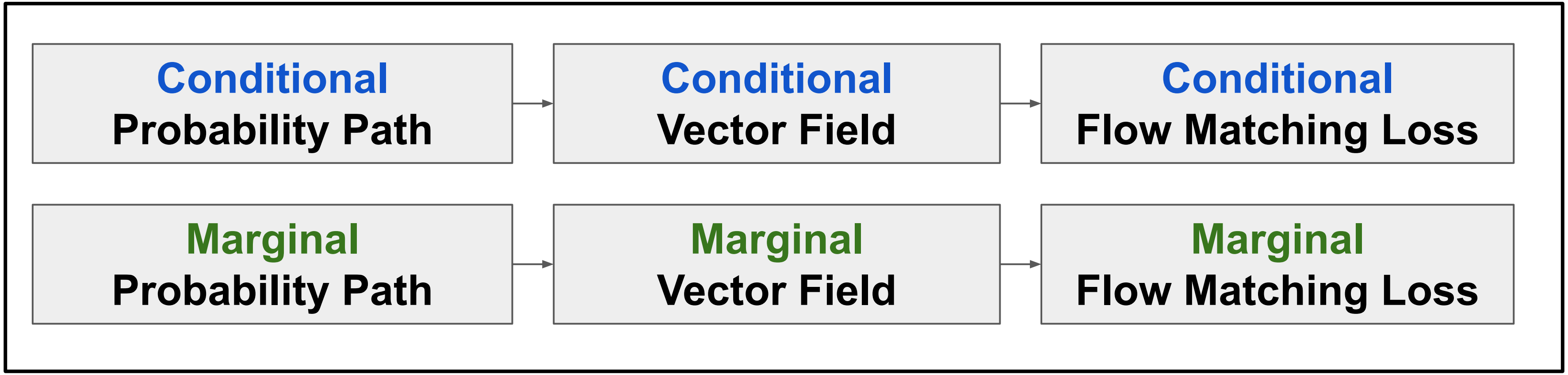
Advantages

- Generate Multiple Tokens in Parallel → More ***Speed?!***
- Generate Tokens in any order → **Text editing?!**
- New probability paths → **Can we design ones that make semantic sense?**

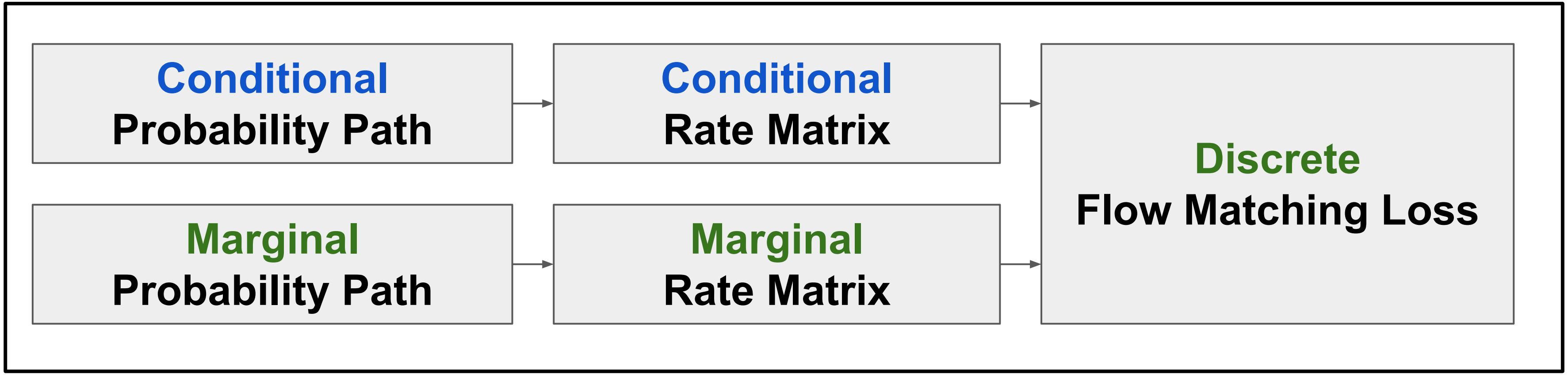
Disadvantages

- No KV caching → **Less *Speed?!***
- Need to learn how to generate Tokens in *any* order → **Harder to learn?!**
- Autoregressive order (left-to-right) makes semantic sense → **Is it worth it?**

Continuous Flow Matching



The Discrete Flow Matching Matrix



How can it be that Flow Matching recipe works so similarly for discrete data and CTMCs?

The principle underlying flow matching is more general. It can be derived for a general class of **Markov processes**:

GENERATOR MATCHING: GENERATIVE MODELING WITH ARBITRARY MARKOV PROCESSES

Peter Holderrieth[1,†], Marton Havasi[2], Jason Yim[1], Neta Shaul[2,3], Itai Gat[2], Tommi Jaakkola[1], Brian Karrer[2], Ricky T. Q. Chen[2], Yaron Lipman[2]

Class Recap

- **Lecture 1 - Flow and Diffusion Models**
- **Lecture 2 - Flow Matching:** Training algorithm.
- **Lecture 3 - Score Matching, Guidance:** How to condition on a prompt.
- **Lecture 4 - Build Image Generators: Latent spaces + Network architectures**
- **Lecture 5 - Discrete diffusion models and flow matching**

This is our final class!

Thank you for joining us!!!!

Please fill out the subject evaluation surveys!