Score Entropy Discrete Diffusion Models
(Lou, Meng, Ermon, ICML 2024)

Aaron Lou, Chenlin Meng, Stefano Ermon

Stanford University

June 13, 2025

Outline

1. Motivation and Background
2. Discrete Diffusion Process
3. Score Entropy: The Key Innovation
4. Training Objective and Implementation
5. Sampling and Generation
6. Empirical Results
7. Summary and Takeaways

Generative Modeling and Diffusion Models

Generative modeling aims to learn the data distribution p_{data} to generate new samples.

Diffusion models (DMs) are a class of generative models that gradually transform data into noise and learn to reverse this process.

Success and Challenges

- DMs excel in continuous domains (images, audio).
- Struggle with discrete data (e.g., text, categorical sequences).
- Discrete domains lack gradients; score matching is nontrivial.

Autoregressive vs. Diffusion for Text

Autoregressive models (e.g., GPT-2) dominate language modeling due to:

- Simple chain rule factorization
- Efficient likelihood computation
- High sample quality

Diffusion models for text:

- Offer parallel generation, controllable infilling
- Historically underperform on likelihoods and sample quality

Discrete Diffusion: The Forward Process

Discrete space: $X = \{1, 2, ..., N\}$ (e.g., vocabulary of size N)
Probability distributions are vectors $p \in \mathbb{R}^N$ with $p_i \geq 0, \sum_i p_i = 1$.

Continuous-time Markov process:

$$\frac{dp_t}{dt} = Q_t p_t, \quad p_0 \approx p_{\text{data}}$$

where Q_t is a transition rate matrix:

- $Q_t(i, j) \geq 0$ for $i \neq j$
- $\sum_i Q_t(i, j) = 0$ (columns sum to zero)

As $t \to \infty$, p_t approaches a simple base distribution p_{base} (e.g., uniform).

Discrete Diffusion: Transition Densities

For small Δt:

$$p(x_{t+\Delta t} = y \mid x_t = x) = \delta_{xy} + Q_t(y, x)\Delta t + O(\Delta t^2)$$

Interpretation:

- With probability $1 - \sum_{y \neq x} Q_t(y, x)\Delta t$, stay at x.
- With probability $Q_t(y, x)\Delta t$, jump from x to y.

Example: For $X = \{A, B, C\}$ and

$$Q = \begin{bmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}$$

the process jumps from any state to another with equal rate.

Reverse Process and Concrete Scores

Reverse diffusion:

$$\frac{dp_{T-t}}{dt} = \tilde{Q}_{T-t} p_{T-t}$$

where

$$\tilde{Q}_t(y, x) = \frac{p_t(y)}{p_t(x)} Q_t(x, y)$$

The ratios $\frac{p_t(y)}{p_t(x)}$ are called **concrete scores** (generalize $\nabla_x \log p_t$). **Goal:**

Learn to approximate these ratios for model-based generation.

Score Matching in Discrete Spaces

Continuous: Score matching learns $\nabla_x \log p(x)$. **Discrete:** Need to learn ratios $\frac{p(y)}{p(x)}$ for $x \neq y$. **Previous approaches:**

- Mean prediction: Learn $p_{0|t}$ (harder, less stable)
- Ratio matching: Maximum likelihood on marginals (expensive)
- Concrete score matching: ℓ_2 loss on ratios (can diverge)

Score Entropy Loss: Definition

Score entropy is a new loss for learning concrete scores:

$$L_{SE} = \mathbb{E}_{x \sim p} \left[\sum_{y \neq x} w_{xy} \left(s_\theta(x)_y - \frac{p(y)}{p(x)} \log s_\theta(x)_y + K\left(\frac{p(y)}{p(x)}\right) \right) \right]$$

where

$$K(a) = a(\log a - 1)$$

and $w_{xy} \geq 0$ are weights (often 1). **Notation:**

- $s_\theta(x)_y$: Model's estimate of $\frac{p(y)}{p(x)}$
- $p(y), p(x)$: True probabilities

Score Entropy: Properties

- **Non-negative, convex, symmetric** (Bregman divergence with $F = -\log$)
- **Consistency:** Minimizing L_{SE} recovers the true ratios.
- **Log-barrier:** Penalizes negative or zero $s_\theta(x)_y$ (keeps outputs positive).
- **Generalizes cross-entropy:** For probabilities, reduces to standard cross-entropy.

Example: If $p(y) = 0.2, p(x) = 0.4,\ s_\theta(x)_y = 0.5$,

$$\frac{p(y)}{p(x)} = 0.5, \quad K(0.5) = 0.5(\log 0.5 - 1) \approx -0.8466$$

Denoising Score Entropy for Diffusion

Practical variant: Use denoising score entropy for scalable training:

$$L_{DSE} = \mathbb{E}_{x_0 \sim p_0, x \sim p(\cdot|x_0)} \left[\sum_{y \neq x} w_{xy} \left(s_\theta(x)_y - \frac{p(y|x_0)}{p(x|x_0)} \log s_\theta(x)_y \right) \right]$$

Interpretation:

- $p(x|x_0)$: Transition probability from x_0 to x after some noise
- $p(y|x_0)$: Transition probability from x_0 to y
- $s_\theta(x)_y$: Model's estimate of their ratio

Sampling: Draw x_0 from data, x from noisy process, compute loss on pairs (x, y).

Likelihood Bound and ELBO

Likelihood training:

$$-\log p_\theta(x_0) \leq L_{DWDSE}(x_0) + D_{KL}(p_{T|0}(\cdot|x_0)\|p_{\textsf{base}})$$

where L_{DWDSE} is the diffusion-weighted denoising score entropy:

$$L_{DWDSE}(x_0) = \int_0^T \mathbb{E}_{x_t \sim p_{t|0}(\cdot|x_0)} \left[\sum_{y \neq x_t} Q_t(x_t, y) \left(s_\theta(x_t, t)_y - \frac{p_{t|0}(y|x_0)}{p_{t|0}(x_t|x_0)} \log$$

This provides an upper bound on negative log-likelihood.

For sequences $x = (x_1, ..., x_d)$, $X = \{1, ..., n\}^d$:

- Use token-level transition matrices Q^{tok}.
- Only perturb one token at a time (sparse Q).
- Score network outputs $s_\theta(x, t) \in \mathbb{R}^{d \times n}$, where $s_\theta(x, t)_{i,y}$ estimates ratio for changing x_i to y.

Transition probabilities:

$$p_{t|0}^{\text{seq}}(\tilde{x}|x) = \prod_{i=1}^{d} p_{t|0}^{\text{tok}}(\tilde{x}_i|x_i)$$

Example: For $d = 3$, $n = 4$:

$$x = (1, 2, 3), \quad \tilde{x} = (1, 4, 3), \quad p_{t|0}^{\text{seq}}(\tilde{x}|x) = p_{t|0}^{\text{tok}}(1|1) \cdot p_{t|0}^{\text{tok}}(4|2) \cdot p_{t|0}^{\text{tok}}(3|3)$$

Transition Matrix Structures

Uniform transitions:

$$Q_{\text{uniform}} = \frac{1}{N}\begin{bmatrix} 1-N & 1 & \cdots & 1 \\ 1 & 1-N & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1-N \end{bmatrix}$$

Absorbing transitions (MASK token):

$$Q_{\text{absorb}} = \begin{bmatrix} -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 0 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}$$

Interpretation: Absorbing state acts like a [MASK] token in BERT.

Simulating the Reverse Diffusion

Goal: Generate x_0 from noise x_T by reversing the diffusion. **Tau-leaping (Euler):** For each token i:

$$p_i(y|x_t^i) = \delta_{x_t^i}(y) + \Delta t Q_t^{\text{tok}}(x_t^i, y) s_\theta(x_t, t)_{i,y}$$

Tweedie denoising:

$$p_{t-\Delta t|t}^{\text{Tweedie}}(x_{t-\Delta t}|x_t) \propto [\exp(-\sigma_t^{\Delta t} Q) s_\theta(x_t, t)_i]_{x_{t-\Delta t}^i} \exp(\sigma_t^{\Delta t} Q)(x_t^i, x_{t-\Delta t}^i)$$

where $\sigma_t^{\Delta t} = \sigma(t) - \sigma(t - \Delta t)$.

Conditional Generation and Infilling

Infilling: Given prompt tokens at positions Ω with values y, generate the rest.

$$\frac{p_t(x_{\bar{\Omega}} = z' | x_\Omega = y)}{p_t(x_{\bar{\Omega}} = z | x_\Omega = y)} = \frac{p_t(x = z' \oplus_\Omega y)}{p_t(x = z \oplus_\Omega y)}$$

Implication: The same score function s_θ can be used for arbitrary conditioning, enabling flexible prompting and infilling.

Language Modeling Results

Perplexity Comparison:

Model	LAMBADA	WikiText2	PTB	1BW
GPT-2 (small)	45.04	42.43	138.43	75.20
SEDD Absorb	$\leq$50.92	$\leq$41.84	$\leq$114.24	$\leq$79.29
SEDD Uniform	$\leq$65.40	$\leq$50.27	$\leq$140.12	$\leq$101.37
D3PM	$\leq$93.47	$\leq$77.28	$\leq$200.82	$\leq$138.92

SEDD outperforms prior diffusion models and is competitive with GPT-2.

Sample Generation Quality

SEDD generates:

- High-quality, coherent text without temperature annealing
- Comparable or better generative perplexity than GPT-2
- Efficient compute-quality tradeoff (fewer network evaluations)

Example: (from paper)

"As Jeff Romer recently wrote, 'The economy has now reached a corner - 64% of household wealth and 80% of wealth goes to credit cards because of government austerity...'"

Summary and Takeaways

- **Score entropy** enables principled, scalable training for discrete diffusion models.
- SEDD achieves state-of-the-art results for non-autoregressive language modeling.
- Enables flexible, parallel, and controllable generation (infilling, arbitrary prompts).
- Bridges the gap between diffusion and autoregressive models for discrete data.

Code: `https://github.com/louaaron/Score-Entropy-Discrete-Diffusion`

References

Lou, A., Meng, C., Ermon, S. (2024). Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. ICML 2024.
https://arxiv.org/abs/2310.16834