Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution

Aaron Lou   Chenlin Meng   Stefano Ermon

Presenter: Keyu Wang

ML Method for Scientific Discovery Seminar 25ws, Dec 17, 2025

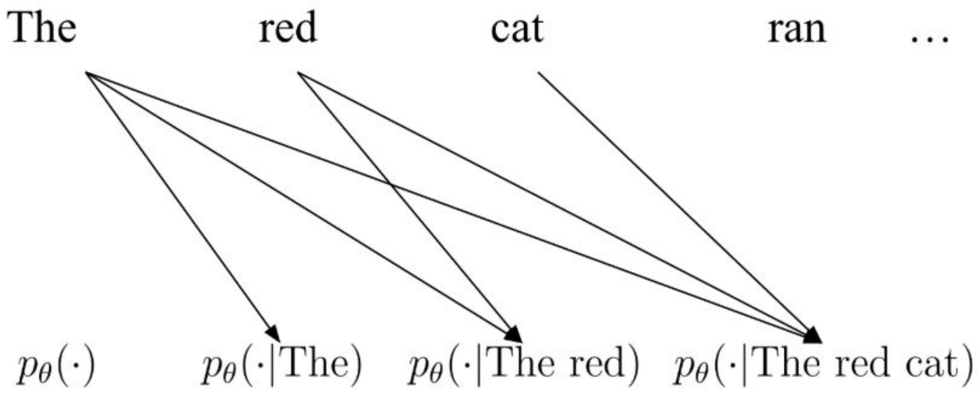
Background: Modeling Discrete Data

Goal: model a distribution over discrete data

$$\mathcal{X} \in \{1, 2, ..., |\mathcal{X}|\} \qquad p_\theta : \mathcal{X} \to \mathcal{R}$$

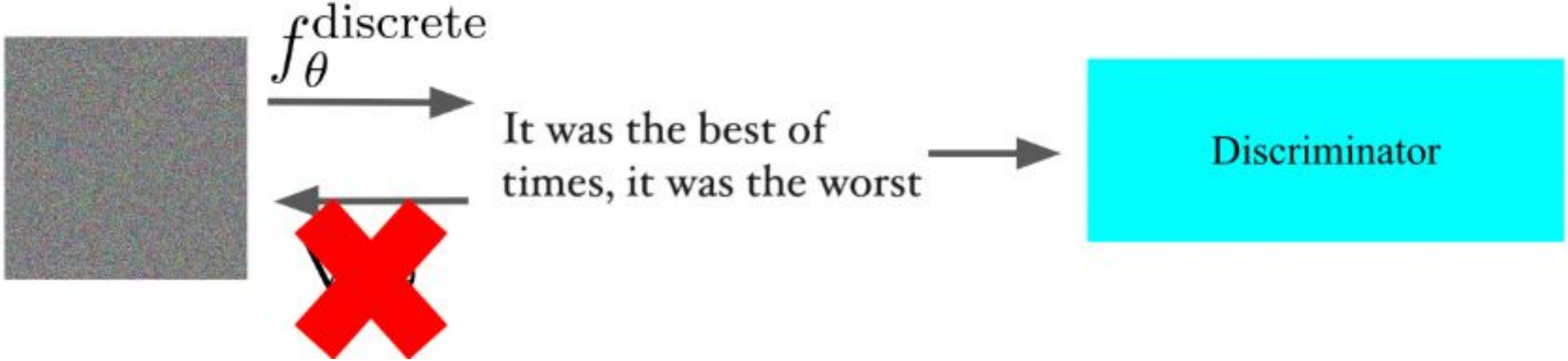
Autoregressive Modeling:

$$p(x) = \prod_{i=1}^{d} p(x_i \mid x_{<i}).$$



Background: Modeling Discrete Data

Autoregressive Modeling **Donimates** (in the past)



Autoregressive Modeling Has **Limits**!

- **Autoregressive generation drift** from the data distribution
- **Sequential sampling** inefficiency on parallel GPUs
- **Limited controllability** from left-to-right generation
  (e.g., infilling given a prefix and a suffix)

Background: Modeling Discrete Data

Any Alternatives?

$$\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$$

$$p_\theta : \mathcal{X} \to \mathbb{R}$$

Rethinking the Challenge:

$$p(x) \geq 0 \quad \text{and} \quad \sum_{x \in \mathcal{X}} p(x) = 1$$

$$p_\theta(x) = \frac{e^{f_\theta(x)}}{Z} \quad \text{where} \quad Z = \sum_{x \in \mathcal{X}} e^{f_\theta(x)}$$

Computing the **partition function Z** is **intractable**!

AAAAAAAAA … AAA
AAAAAAAAA … AAB
AAAAAAAAA … AAC

……………………………

……………………………

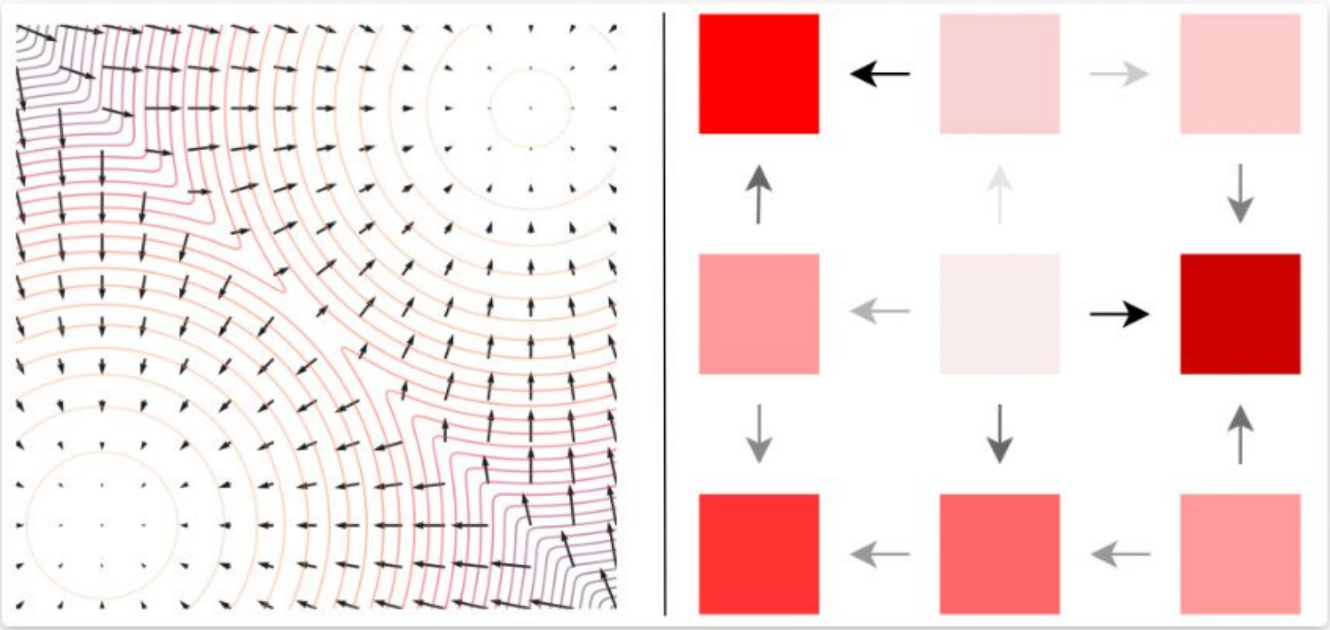
……………………………

ZZZZZZZZZ … ZZY
ZZZZZZZZZ … ZZZ

Summing over all possible sequences is computationally impossible.

Related Work: Concrete Scores

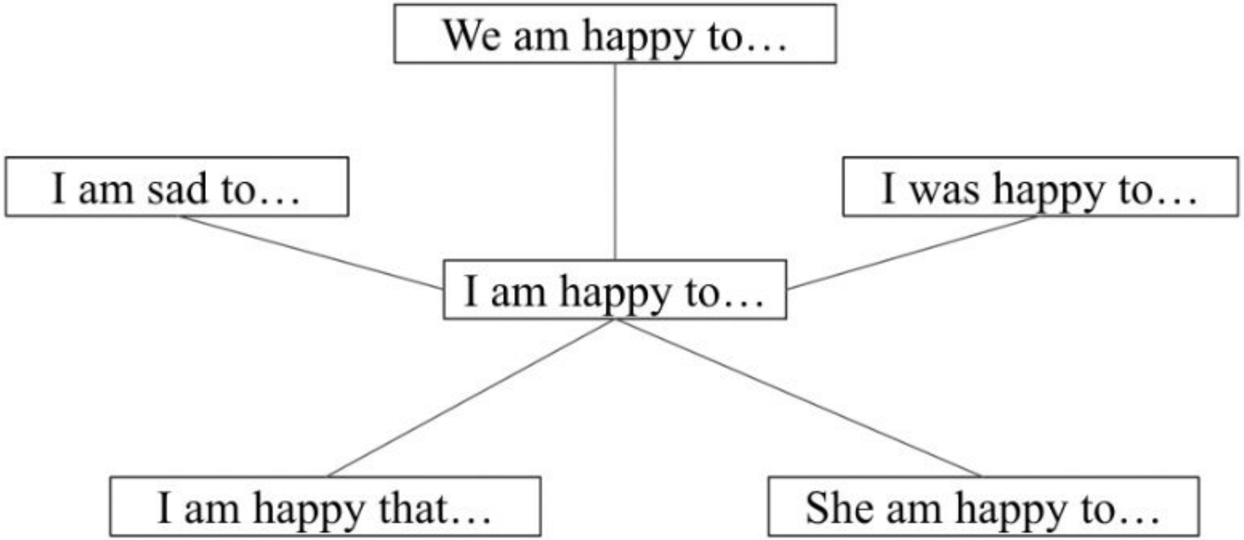
$$s_\theta(x)_y \qquad \frac{p_\theta(y)}{p_\theta(x)} = \frac{e^{f_\theta(y)}/Z}{e^{f_\theta(x)}/Z} = \frac{e^{f_\theta(y)}}{e^{f_\theta(x)}}$$



$\nabla_x \log p$ $\qquad$ $\frac{p_t(y)}{p_t(x)}$

Related Work: Concrete Scores

Real Practice



model the probability ratios between neighboring sequences,
that differ at exactly one position.

For text, this means comparing sentences where only one token is changed!

Related Work: Concrete Scores

Concrete Score Matching (Meng et al. 2022)

$$\mathcal{L}_{\mathrm{CSM}} = \frac{1}{2}\mathbb{E}_{x\sim p}\left[\sum_{y\neq x}\left(s_\theta(x)_y - \frac{p(y)}{p(x)}\right)^2\right].$$

Not sufficiently **penalize negative or zero values**, leading to divergent behavior!

Method: Score Entropy Discrete Diffusion

Score Entropy:

$$\sum_{y \sim x} s_\theta(x)_y - \frac{p_{\text{data}}(y)}{p_{\text{data}}(x)} \log s_\theta(x)_y$$

Explanation:

$$f(s) = s - a \log s, \qquad a = \frac{p_{\text{data}}(y)}{p_{\text{data}}(x)} > 0, \; s > 0.$$

2nd derivative : $$f''(s) = \frac{a}{s^2} > 0 \quad \forall s > 0.$$

1st derivative : $$f'(s) = 1 - \frac{a}{s}. \qquad s^\star = a = \frac{p_{\text{data}}(y)}{p_{\text{data}}(x)}.$$

Method: Score Entropy Discrete Diffusion

Real Practice:

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

Forward diffusion process

$$\frac{dp_t}{dt} = Qp_t \qquad p_0 = p_{\text{data}}$$

$$p(x_{t+\Delta t} = a | x_t = b) \approx \delta_b(a) + Q(a,b)\Delta t$$

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

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

[MASK]

Method: Score Entropy Discrete Diffusion

Generating Samples: Reverse diffusion process

$$\frac{dp_{T-t}}{dt} = \overline{Q}_{T-t} p_{T-t} \quad \overline{Q}_t(x, y) = \frac{p_t(y)}{p_t(x)} Q(y, x)$$

$$p(x_{t-\Delta t} = a | x_t = b) \approx \delta_b(a) + \frac{p_t(a)}{p_t(b)} Q(b, a) \Delta t$$

Method: Score Entropy Discrete Diffusion

Summary

Algorithm 1 Score Entropy Training Loop (Multiple Dimensions)

Require: Network s_θ, noise schedule σ (total noise $\overline{\sigma}$), data distribution p_{data}, token transition matrix Q, time $[0, T]$.

Sample $\mathbf{x}_0 \sim p_0$, $t \sim \mathcal{U}([0, T])$.

Construct $\mathbf{x}_t$ from $\mathbf{x}_0$. In particular, $x_t^i \sim p_{t|0}(\cdot | x_0^i) = \exp(\overline{\sigma}(t) Q)_{x_0^i}$.

if Q is Absorb **then**

This is $e^{-\overline{\sigma}(t)} e_{x_0^i} + \left(1 - e^{-\overline{\sigma}(t)}\right) e_{\text{MASK}}$

else if Q is Uniform **then**

This is $\frac{e^{\overline{\sigma}(t)} - 1}{n e^{\overline{\sigma}(t)}} \mathbb{1} + e^{-\overline{\sigma}(t)} e_{x_0^i}$

end if

Compute $\widehat{\mathcal{L}}_{DWDSE} = \sigma(t) \sum_{i=1}^{d} \sum_{y=1}^{n} (1 - \delta_{x_t^i}(y)) \left(s_\theta(\mathbf{x}_t, t)_{i,y} - \frac{p_{t|0}(y | x_0^i)}{p_{t|0}(x_t^i | x_0^i)} \log s_\theta(\mathbf{x}_t, t)_{i,y} \right)$.

Backpropagate $\nabla_\theta \widehat{\mathcal{L}}_{DWDSE}$. Run optimizer.

Method: Score Entropy Discrete Diffusion

Summary

Algorithm 2 Score Entropy Sampling (Unconditional)

Require: Network s_θ, noise schedule σ (total noise $\overline{\sigma}$), token transition matrix Q, time $[0, T]$, step size Δt

Sample $\mathbf{x}_T \sim p_{\text{base}}$ by sampling each x_T^i from the stationary distribution of Q.

$t \leftarrow T$

while $t > 0$ **do**

  if Using Euler **then**

    Construct transition densities $p^i(y|x_t^i) = \delta_{x_t^i}(y) + \Delta t Q_t^{\text{tok}}(x_t^i, y) s_\theta(\mathbf{x}_t, t)_{i,y}$.

  else if Using Tweedie Denoising **then**

    Construct transition densities $p^i(y|x_t^i) = \left(\exp(\overline{\sigma}(t - \Delta t) - \overline{\sigma}(t))Q) s_\theta(\mathbf{x}_t, t)_i\right)_y \exp((\overline{\sigma}(t) - \overline{\sigma}(t - \Delta t))Q)(x_t^i, y)$

  end if

  Normalize $p^i(\cdot|x_t^i)$ (clamp the values to be minimum 0 and renormalize the sum to 1 if needed).

  Sample $x_{t-\Delta t}^i \sim p^i(y|x_t^i)$ for all i, constructing $\mathbf{x}_{t-\Delta t}$ from $x_{t-\Delta t}^i$.

  $t \leftarrow t - \Delta t$

end while

Return: $\mathbf{x}_0$

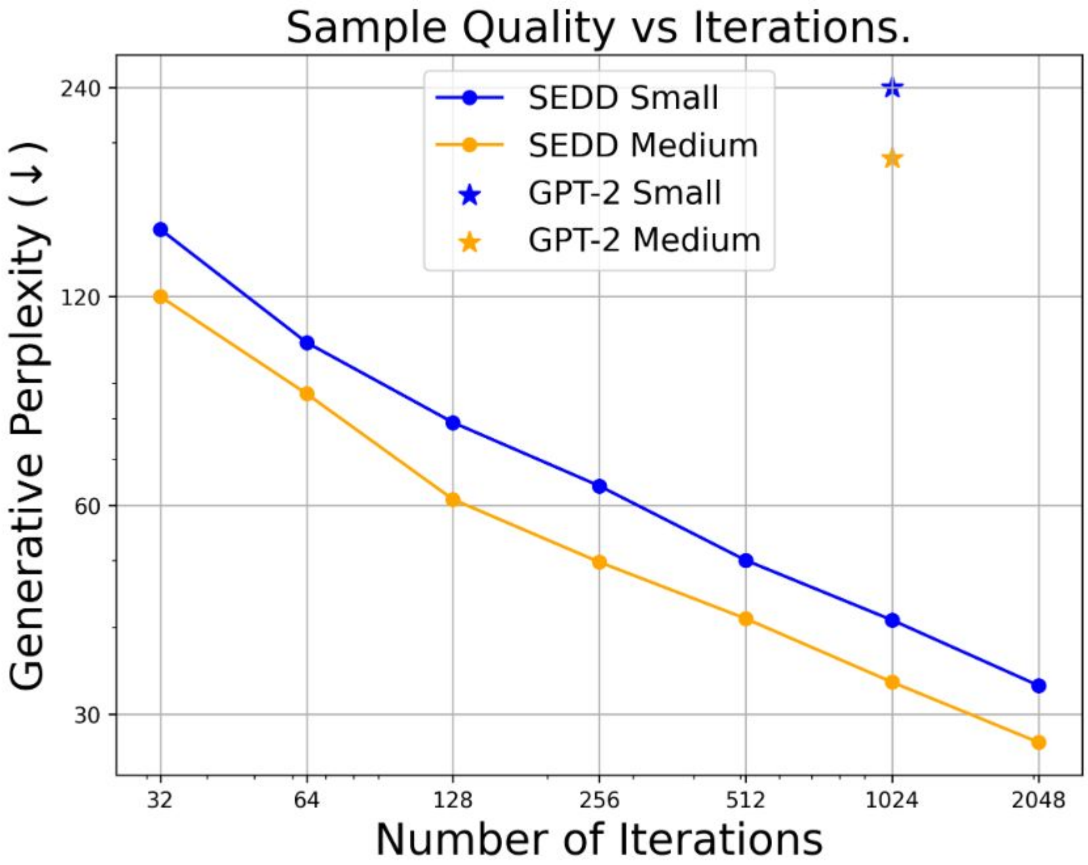
Evaluation: sampling quality

Method	LAMBADA PPL	Wikitext2 PPL	PTB PPL	Wikitext103 PPL	1BW PPL
GPT-2 Small	**45.04**	42.43	138.43	41.60	**75.20**
SEDD Small	≤50.92	**≤41.84**	**≤114.24**	**≤40.62**	≤79.29
GPT-2 Medium	**35.66**	31.80	123.14	31.39	**55.72**
SEDD Medium	≤42.77	**≤31.04**	**≤87.12**	**≤29.98**	≤61.19

Evaluation: sampling quality

GPT-2 S	Fantagraphics Phoenix's National of F***ey Fanful in danger notwithstanding, even with her toxicity everywhere the kitten roller's nastiness and absence are recycled by Atlas Shrugged fandom. 21 pages from such energetic and arguably beautifully framed hand drawn art is convincing points in the beaded and twisted edges of immatainer on the one hand, chocolate milk prescription and misadventure narrative
GPT-2 M	editQuotedCrons Kungfu Forbidden Comics series fixtures for retro gaming You might need a graphic graphic display/two-dimensional window operation. If you do not have a slope or column plane display what return jpeg could you switch between adjust pyg in geault window? Will I be bothered if I see Hikaru If Disney gets a lawsuit involving best selling romance novels, in some country with this crew
SEDD S	As a director, I am not really involved since it's completely collaborative. Instead of me directing writers and what they say, this show is going to tell me a whole lot of different things. But it is not about "not on the show," I'll do scenes and the script will be left to the writer. So I won't really care about the final act of the episode or even the ending. All the premises of the show are going to be done to experiment
SEDD M	Aar Heeich, explained Google plans in remarks at the I/O conference. The video shows just how compatible Google Chrome seems to work with every version of Google's version of Android. Android is the only platform to ship Chrome on nearly every version of its operating system, but Microsoft recently said that it wants to make the replacement app available alongside future versions of Android. The new browser

Evaluation: trade off compute for quality



Evaluation: Flexible Prompt Schemes

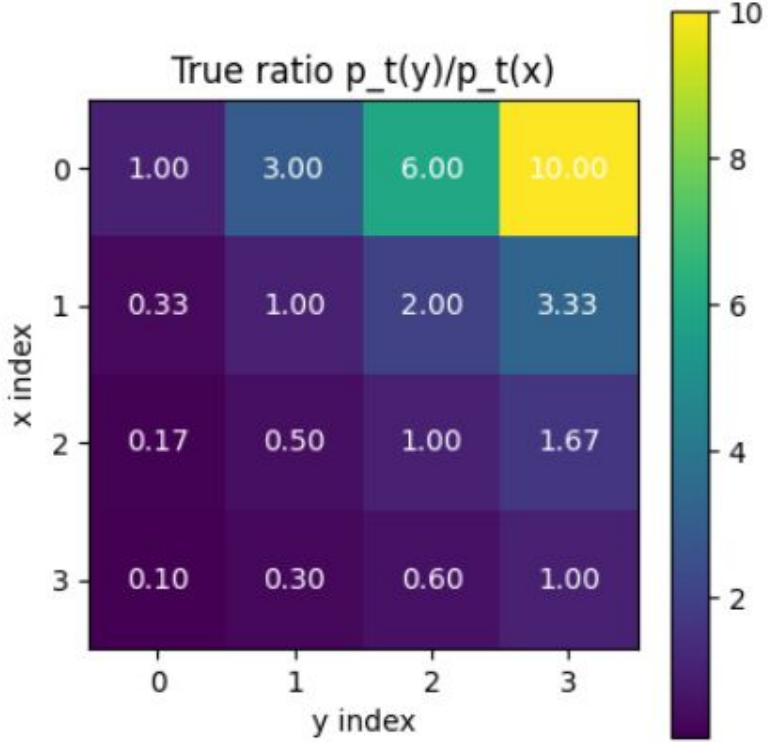
Given an input **c** at indices **Ω** and wish to fill in the remaining indices

$$\frac{p_t(\overline{\Omega} = z \mid \Omega = c)}{p_t(\overline{\Omega} = y \mid \Omega = c)} = \frac{p_t(\overline{\Omega} = z \cap \Omega = c)/p_t(\Omega = c)}{p_t(\overline{\Omega} = y \cap \Omega = c)/p_t(\Omega = c)} = \frac{p_t(\overline{\Omega} = z \cap \Omega = c)}{p_t(\overline{\Omega} = y \cap \Omega = c)}$$

Method	MAUVE Score (↑)
GPT-2 w/ nucleus sampling	0.955
SEDD w/ standard prompting	**0.957**
SEDD w/ infilling prompting	0.942

Toy Demo I

```python
p = torch.tensor([0.05, 0.15, 0.30, 0.50])
p = p / p.sum()
true_ratio = p.view(1, 4) / p.view(4, 1)
```




```python
class RatioNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(8, 32),
            nn.ReLU(),
            nn.Linear(32, 1)
        )

    def forward(self, x_idx, y_idx):
        x_oh = F.one_hot(x_idx, num_classes=4).float()    # [B, 4]
        y_oh = F.one_hot(y_idx, num_classes=4).float()    # [B, 4]
        inp = torch.cat([x_oh, y_oh], dim=-1)             # [B, 8]
        out = self.net(inp).squeeze(-1)                   # [B]
        return out
```

Goal: Predict True Ratio

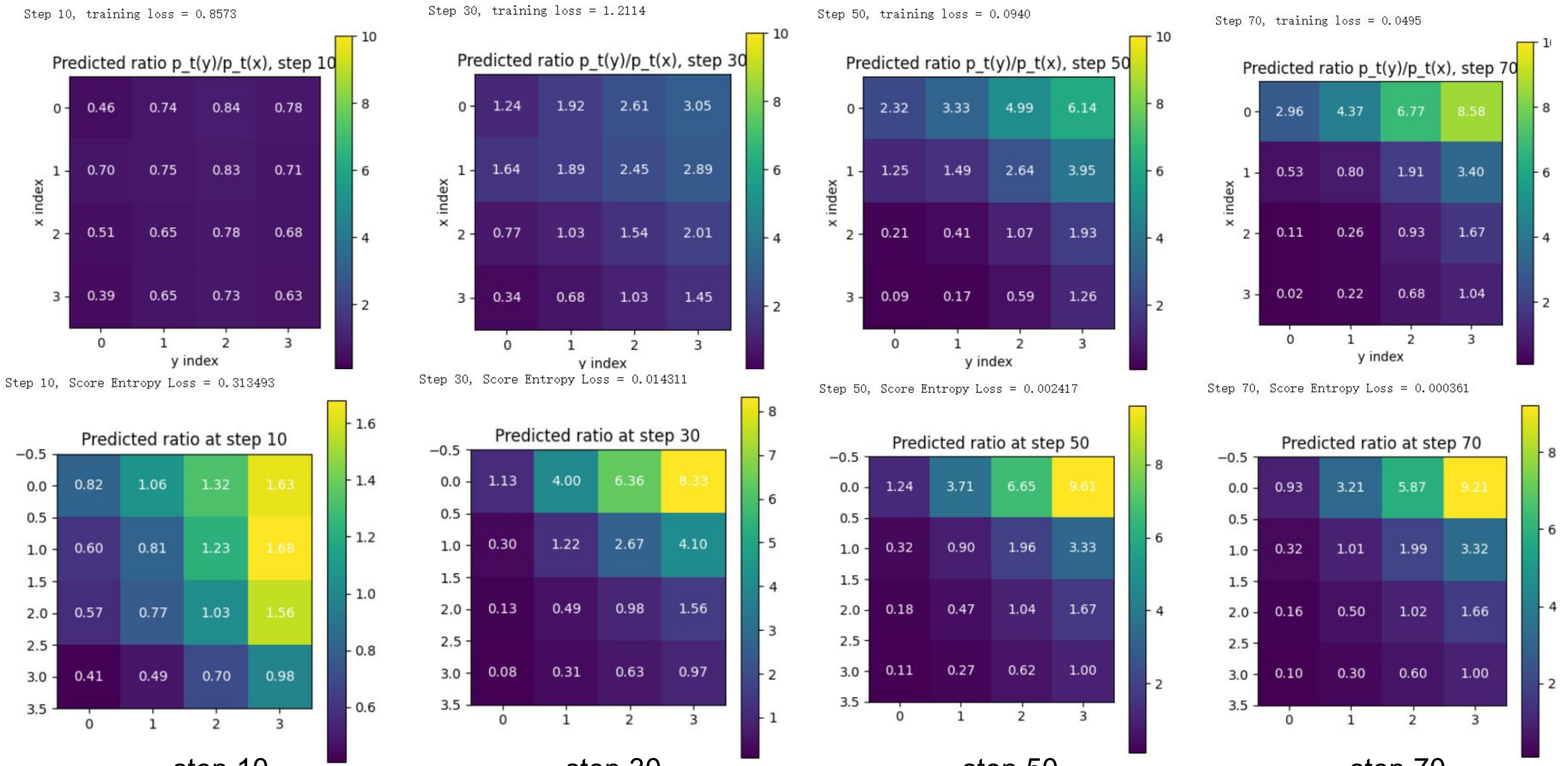
CSM(MSE Loss)
v.s. SEDD(Score Entropy Loss)

Toy Demo I

SEDD converges much faster than CSM!

MSE

Score Entropy



step 10 step 30 step 50 step 70

Toy Demo II

```python
# Toy sentence dataset: each sequence has fixed length 4
sentences = [
    ["I",   "love", "AI",  "."],
    ["You", "love", "AI",  "."],
    ["We",  "love", "AI",  "."],
    ["I",   "love", "NLP", "."],
    ["You", "love", "NLP", "."],
    ["We",  "love", "NLP", "."],
]

seq_len = 4

# Explicitly include [MASK] token in the vocabulary
vocab = ["", "I", "You", "We", "love", "AI", "NLP", ".", "[MASK]"]
token2id = {tok: i for i, tok in enumerate(vocab)}
id2token = {i: tok for tok, i in token2id.items()}
MASK_ID = token2id["[MASK]"]
vocab_size = len(vocab)

print("vocab_size =", vocab_size)
print("MASK_ID    =", MASK_ID)

def encode_sentence(tokens):
    assert len(tokens) == seq_len
    return torch.tensor([token2id[t] for t in tokens], dtype=torch.long)

data_x0 = torch.stack([encode_sentence(s) for s in sentences]).to(device)  # (N, L)
num_samples = data_x0.size(0)
```

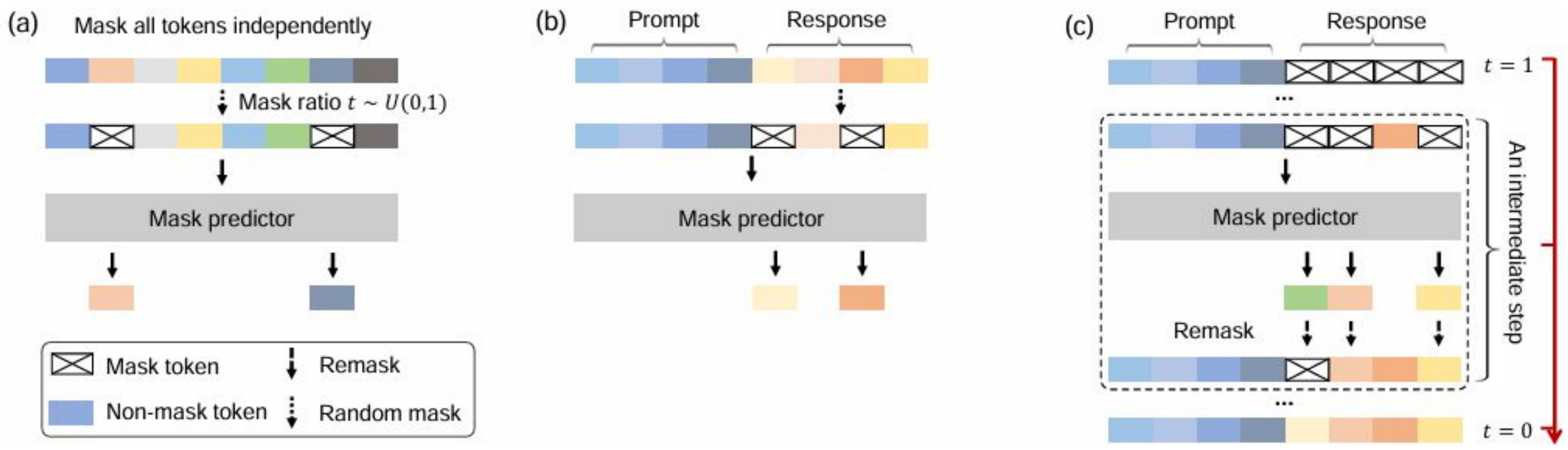
```python
class SimpleSEDDModel(nn.Module):
    def __init__(self, vocab_size, d_model, T, seq_len):
        super().__init__()
        self.token_emb = nn.Embedding(vocab_size, d_model)
        self.pos_emb   = nn.Embedding(seq_len, d_model)
        self.time_emb  = nn.Embedding(T + 1, d_model)  # t ∈ [1..T], index 0 unused

        self.mlp = nn.Sequential(
            nn.Linear(d_model, d_model),
            nn.GELU(),
            nn.Linear(d_model, vocab_size),
        )

        self.register_buffer("positions", torch.arange(seq_len).long())

    def forward(self, x_t, t):
        """
        x_t: (B, L)
        t  : (B,)
        Returns logits: (B, L, V)
        """
        B, L = x_t.shape

        tok_emb = self.token_emb(x_t)                        # (B, L, D)
        pos_emb = self.pos_emb(self.positions)[None, :, :]   # (1, L, D)
        time_emb = self.time_emb(t).unsqueeze(1)             # (B, 1, D)

        h = tok_emb + pos_emb + time_emb                     # (B, L, D)
        logits = self.mlp(h)                                 # (B, L, V)
        return logits

model = SimpleSEDDModel(vocab_size=vocab_size, d_model=64, T=T, seq_len=seq_len).to(device)
```

Toy Demo II

Clean Sentence	Noised Input	Prediction
We love NLP .	We AI love You	We love NLP .
We love NLP .	We love You .	We love NLP .
You love NLP .	You . NLP .	You love NLP .
I love AI .	I love AI You	I love AI .
You love NLP .	love AI love	I love AI .
I love NLP .	We I .	We love NLP .
I love AI .	We love We	We love AI .
I love AI .	. love love I	We love AI .
I love AI .	love AI You .	I love NLP .
You love AI .	love NLP AI love	We love AI .

Input x_t	Prediction
[MASK] [MASK] [MASK] [MASK]	I love NLP .
[MASK] [MASK] [MASK] [MASK]	We love NLP .
[MASK] [MASK] [MASK] [MASK]	You love NLP .
[MASK] [MASK] [MASK] [MASK]	I love NLP .
[MASK] [MASK] [MASK] [MASK]	You love AI .
[MASK] [MASK] [MASK] [MASK]	We love AI .
[MASK] [MASK] [MASK] [MASK]	You love AI .
[MASK] [MASK] [MASK] [MASK]	I love AI .
[MASK] [MASK] [MASK] [MASK]	You love AI .
[MASK] [MASK] [MASK] [MASK]	I love NLP .

Discussion: SEDD v.s. Masked Discrete Diffusion (MDD)



MDD learns language representations by randomly masking tokens and training the model to predict the masked tokens.

distribution geometry v.s. token reconstruction
theoretical supports v.s. more stable in practice

Conclusion

- Stable Ratio Estimation

- Unified Training Objective

- Improved Sampling Efficiency

- Nice Theory for Discrete Domains

References

[1] Score Entropy Discrete Diffusion (SEDD):

Lou A, Meng C, Ermon S. Discrete diffusion modeling by estimating the ratios of the data distribution[J]. Proceedings of the 41st International Conference on Machine Learning, 2024.

[2] Concrete Score Matching (CSM):

Meng C, Choi K, Song J, et al. Concrete score matching: Generalized score matching for discrete data[J]. Advances in Neural Information Processing Systems, 2022.

Thanks

Questions?