Carnegie Mellon University

Lecture 12: Discrete Diffusion & Masked Diffusion

Yutong (Kelly) He

10-799 Diffusion & Flow Matching, Feb 5th, 2026

Modal aws

Quiz time!

10 minutes

Closed-book

Pen & Paper

If you don’t want to stay for the lecture, feel free to leave after submitting your quiz!

Housekeeping Announcements

- Homework 4 is out! https://kellyyutonghe.github.io/10799S26/homework/
    - Due date: 2/27 Fri, Late Due date: 3/1 Sun
- Poster session:
    - PDF submission 2/25 Wed
    - Poster Session 2/26 Thur 5 PM tot 7 PM, same classroom
- No class on 2/24 Tue

So far we have learned about (almost) everything about diffusion models for image generation

Fundamentals:

- Denoising diffusion models
- Score-based models
- Flow matching

Advanced topics:

- The design space
- Fast sampling solvers
- Controllable generations
- Text-to-image generations
- Distillation & Self distillation

So far everything we have learned are about image diffusion (i.e. in a continuous space)

Fundamentals:

- Denoising diffusion models for image
- Score-based models for image
- Flow matching for image

Advanced topics:

- The design space for image diffusion
- Fast sampling solvers for continuous ODE
- Controllable generations for image diffusion
- Text-to-image generations
- Distillation & Self distillation for image diffusion

So far everything we have learned are about image diffusion (i.e. in a continuous space)

Fundamentals:

- Denoising diffusion models for image
- Score-based models for image
- Flow matching for image

Advanced topics:

- The design space for image diffusion
- Fast sampling solvers for continuous ODE
- Controllable generations for image diffusion
- Text-to-image generations
- Distillation & Self distillation for image diffusion

How about discrete data?

Fundamentals:

- Denoising diffusion models for image
- Score-based models for image
- Flow matching for image

Advanced topics:

- The design space for image diffusion
- Fast sampling solvers for continuous ODE
- Controllable generations for image diffusion
- Text-to-image generations
- Distillation & Self distillation for image diffusion

Well discrete data is very important

Carnegie Mellon University

Images from https://openai.com/index/introducing-chatgpt-search/, https://news.osu.edu/how-earths-molecules-got-their-handedness/, https://www.ultimate-photo-tips.com/what-is-a-pixel.html

How to make diffusion models work on discrete data

Fundamentals:

- Denoising diffusion models ~~for image~~ text, molecules …
- Score-based models ~~for image~~ text, molecules …
- Flow matching ~~for image~~ text, molecules …

Diffusion models can be viewed in three ways

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise

- Score-based models

  => Learning the score function

- Flow matching

  => Learning the velocity

Discrete Diffusion: ?

Let’s look at them one by one

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise

- Score-based models

  => Learning the score function

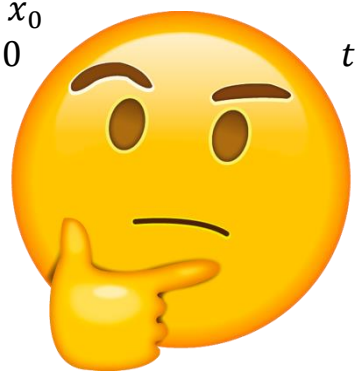
- Flow matching

  => Learning the velocity

Diffusion's way to add noise and denoise

Forward process (adding noise)

Data x_0 $t = 0$ | x_1 $t = 1$ | x_2 $t = 2$ | x_3 $t = 3$ | Noise x_4 $t = 4 = T$



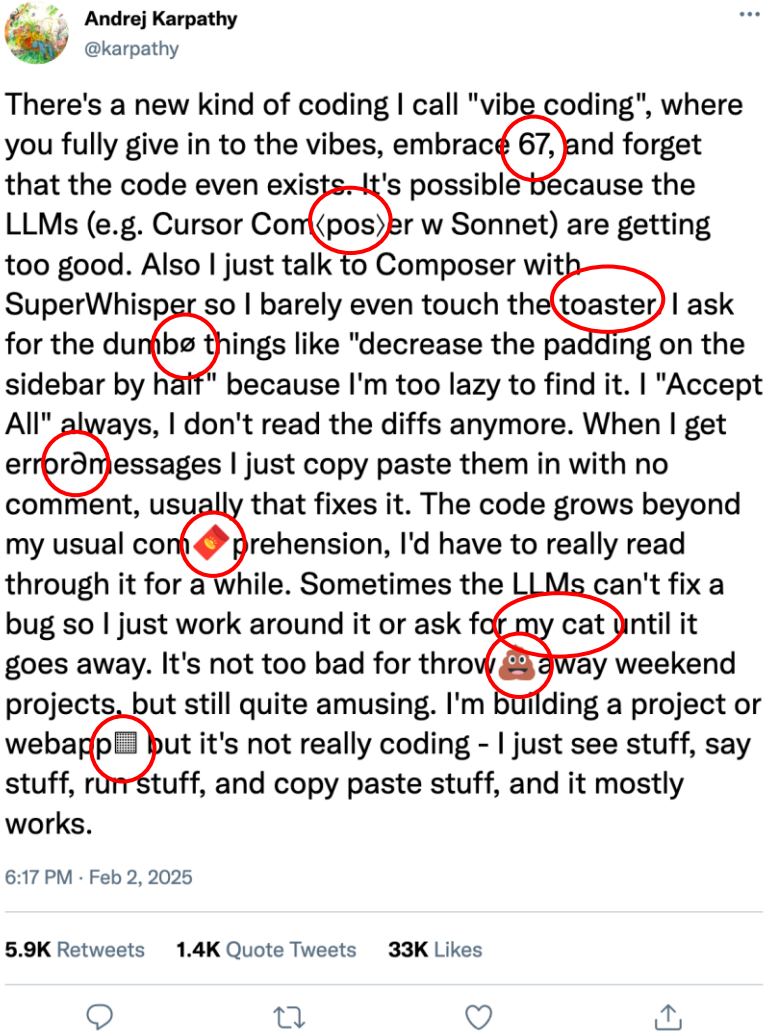
Reserve process (denoising)

Carnegie Mellon University

Cat stolen from Chieh-Hsin (Jesse) Lai

What is noise in text

Identify the “noise” in this tweet:



Carnegie Mellon University

How to add noise in text

Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

Copy Thread

6:17 PM · Feb 2, 2025 · **6.7M** Views

1.4K 5.9K 33K 17K

Image from Spiderman 2002

Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace 67, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Com⟨pos⟩er w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the toaster I ask for the dumbø things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get errorðmessages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual com🧧prehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for my cat until it goes away. It's not too bad for throw💩away weekend projects, but still quite amusing. I'm building a project or webapp▩ but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

6:17 PM · Feb 2, 2025

5.9K Retweets **1.4K** Quote Tweets **33K** Likes

Idea: every token has some probability of getting transformed into a random one

Carnegie Mellon University

How to add noise to text

Let's say we have vocab {I, love, cat}

Then the sentence "I love cat" can be represented by 3 one-hot vectors:

I: [1,0,0,0,0], love: [0,1,0,0,0], cat: [0,0,1,0,0]

Say we have β chance to turn an existing token into a random one in the vocab, then this transformation can be represented by this transition matrix:

$$Q = \begin{bmatrix} 1 - \frac{2\beta}{3} & \frac{\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & 1 - \frac{2\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & \frac{\beta}{3} & 1 - \frac{2\beta}{3} \end{bmatrix}$$

How to add noise to text

Now apply this transition matrix to the third token “cat” to get the categorical distribution that we are sampling from next:

$$x_{\text{cat}}Q = [0 \;\; 0 \;\; 1]\begin{bmatrix} 1-\frac{2\beta}{3} & \frac{\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & 1-\frac{2\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & \frac{\beta}{3} & 1-\frac{2\beta}{3} \end{bmatrix} = [\frac{\beta}{3} \quad \frac{\beta}{3} \quad 1-\frac{2\beta}{3}]$$

Probability of getting transitioned into “I”

Probability of getting transitioned into “love”

Probability of staying at “cat”

How to add noise to text

Now apply this transition matrix to the third token “cat” to get the categorical distribution that we are sampling from next:

$$x_{\text{cat}}Q = [0 \;\; 0 \;\; 1]\begin{bmatrix} 1-\frac{2\beta}{3} & \frac{\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & 1-\frac{2\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & \frac{\beta}{3} & 1-\frac{2\beta}{3} \end{bmatrix} = [\frac{\beta}{3} \quad \frac{\beta}{3} \quad 1-\frac{2\beta}{3}]$$

The categorical probability of the transformed token is

$$x_{\text{transformed}}|x_{\text{cat}} \sim \text{Cat}(p = x_{\text{cat}}Q)$$

We can stack all three tokens up and independently apply Q to each and get

How to add noise to text

We can stack all three tokens up to get :

$$x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and independently apply Q to each and get

$$xQ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1-\frac{2\beta}{3} & \frac{\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & 1-\frac{2\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & \frac{\beta}{3} & 1-\frac{2\beta}{3} \end{bmatrix} = \begin{bmatrix} 1-\frac{2\beta}{3} & \frac{\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & 1-\frac{2\beta}{3} & \frac{\beta}{3} \\ \frac{\beta}{3} & \frac{\beta}{3} & 1-\frac{2\beta}{3} \end{bmatrix}$$

The categorical probability of the transformed token is

$$x'|x \sim \text{Cat}(p = xQ)$$

Carnegie Mellon University

Now let's build the diffusion forward process with this

Say we have clean data x_0, and at time t we have β chance to turn an existing token into a random one in the vocab, then this transformation can be represented by this transition matrix

$$Q_t = \begin{bmatrix} 1 - \frac{2\beta_t}{3} & \frac{\beta_t}{3} & \frac{\beta_t}{3} \\ \frac{\beta_t}{3} & 1 - \frac{2\beta_t}{3} & \frac{\beta_t}{3} \\ \frac{\beta_t}{3} & \frac{\beta_t}{3} & 1 - \frac{2\beta_t}{3} \end{bmatrix}$$

The categorical probability of the transformed token is

$$x_t | x_{t-1} \sim \text{Cat}(p = x_{t-1} Q_t)$$

Now let's build the diffusion forward process with this

The categorical probability of the transformed token is

$$x_t|x_{t-1} \sim \text{Cat}(p = x_{t-1}Q_t)$$

By induction, we can get the categorical probability of transforming from t-2 to t

$$x_t|x_{t-2} \sim \text{Cat}(p = x_{t-2}Q_{t-1}Q_t)$$

…

Then we can also get the probability transforming from 0 to t

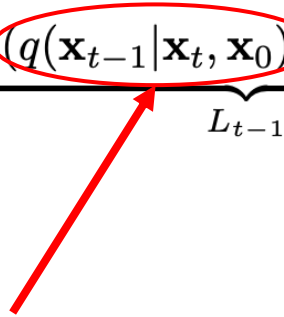
$$x_t|x_0 \sim \text{Cat}(p = x_0\bar{Q}_t)$$

Where $\bar{Q}_t = Q_1Q_2 \dots Q_t$

How to train the reverse process

Remember how in DDPM we have our ELBO

$$\mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$



Pretty much only

need to deal with this

Let's get our training target

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

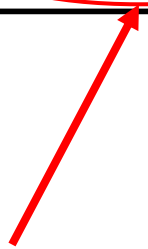
$$= \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$$= \frac{(x_{t-1}Q_t x_t^\top)x_0\bar{Q}_{t-1}}{x_0\bar{Q}_t}$$

$$= x_{t-1}\frac{x_t Q_t^\top \odot x_0\bar{Q}_{t-1}}{x_0\bar{Q}_t}$$

Remember how in DDPM we have our ELBO

$$\mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$



Better parameterization

of this?

Reparameterization trick in discrete diffusion

$$p(x_{t-1}|x_t) = \sum_{x_0} p(x_{t-1}, x_0|x_t) = \sum_{x_0} p(x_{t-1}, x_t|x_0)\ p(x_0|x_t)$$

$$\Rightarrow p_\theta(x_{t-1}|x_t) \approx \sum_{x_0} q(x_{t-1}, x_t|x_0)\ p_\theta(x_0|x_t)$$

Only need to predict the logits

of the final clean output

Pro tips: You can also add another cross entropy loss to directly predict from t to 0: $\mathbb{E}_{q(\boldsymbol{x}_0)}\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}[-\log \widetilde{p}_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)]$.

Carnegie Mellon University

Putting everything together, we got discrete denoising diffusion models (D3PM)

Structured Denoising Diffusion Models in Discrete State-Spaces

Jacob Austin,* Daniel D. Johnson,* Jonathan Ho, Daniel Tarlow & Rianne van den Berg†
Google Research, Brain Team
{jaaustin,ddjohnson,jonathanho,dtarlow,riannevdberg}@google.com

Carnegie Mellon University

Let’s look at them one by one

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise ✅

- Score-based models

  => Learning the score function

- Flow matching

  => Learning the velocity

Discrete Diffusion:

- Discrete denoising diffusion models

  => Categorical noise

- ?

- ?

Let’s look at them one by one

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise ✅

- Score-based models

  => Learning the score function

- Flow matching

  => Learning the velocity

Discrete Diffusion:

- Discrete denoising diffusion models

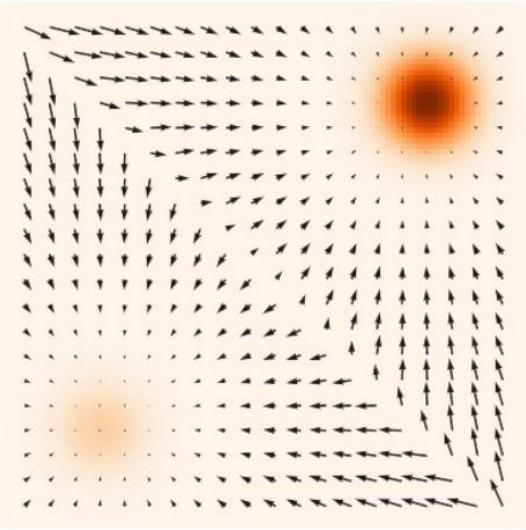
  => Categorical noise

- ?

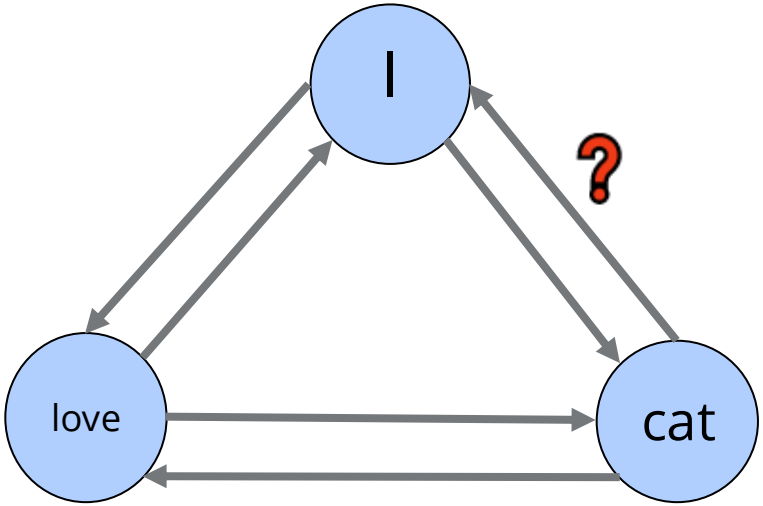
- ?

The continuous score v.s. the “discrete score”

Continuous score:

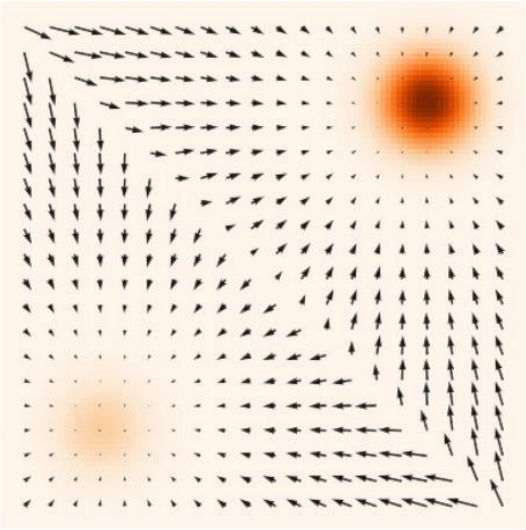


“Discrete score”:



The continuous score v.s. the “discrete score”

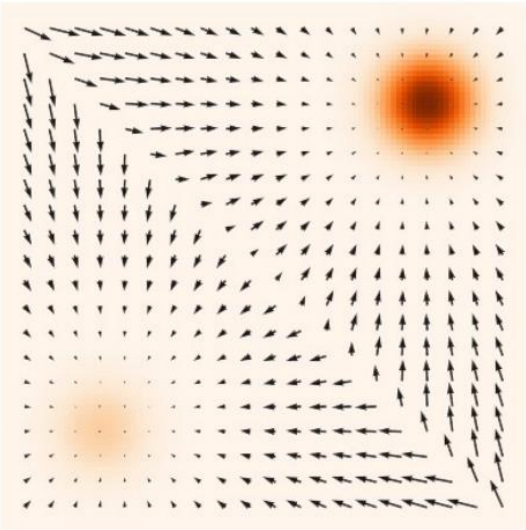
Continuous score:



“Compare my likelihood with my neighbors, if they have higher likelihood than me, I **flow** to them”

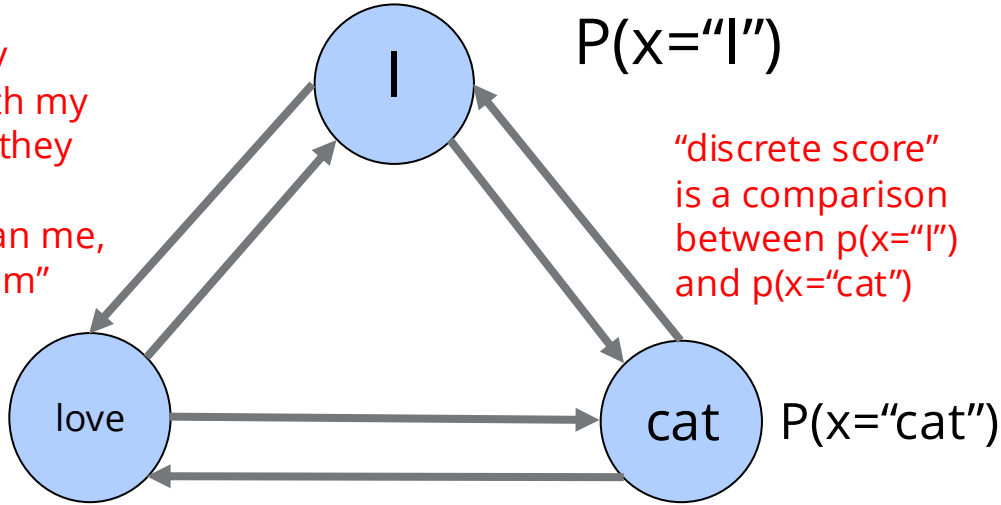
The continuous score v.s. the “discrete score”

Continuous score:



“Compare my likelihood with my neighbors, if they have higher likelihood than me, I **jump** to them”

“Discrete score”:



“discrete score” is a comparison between p(x=“I”) and p(x=“cat”)

Concrete score: $s_\theta(x,t)_y \approx \frac{p_t(y)}{p_t(x)}$

Carnegie Mellon University

Meng et al. Concrete Score Matching: Generalized Score Matching for Discrete Data. NeurIPS 2022. https://arxiv.org/pdf/2211.00802

The continuous time setting

In continuous score-based models, we need to represent the sample evolution in a **continuous time** SDE/ODE

=> How to use continuous time to represent these discrete jumps?

Continuous time Markov Chain (CTMC)

Let's still use Q_t to represent the transition matrix at time t, and the probability to jump from state x to the next infinitesimal step can be written as

$$p_{t+dt}(y|x_t) = \begin{cases} Q_t(x,y)dt & \text{for } x \neq y \\ 1 - \sum_{z \neq x} Q_t(x,z)dt & \text{for } x = y \text{ (i.e. } Q_t(x,x) = -\sum_{z \neq x} Q_t(x,z)) \end{cases}$$

$\Rightarrow$ We can also write this continuous time evolution into an ODE

$$\frac{dp_t}{dt} = Q_t p_t$$

In fact, going in reverse $\frac{dp_{T-t}}{dt} = \overline{Q}_{T-t} p_{T-t}$

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

$$\overline{Q}_t(x,x) = -\sum_{y \neq x} \overline{Q}_t(y,x)$$

Concrete score

Reverse transition

Carnegie Mellon University

Concrete score matching

Then now all we need to do is to match to the concrete score (or do we?)

Concrete Score Matching. Meng et al. (2022) generalizes the standard Fisher divergence in score matching, learning $s_\theta(x,t) \approx \left[\frac{p_t(y)}{p_t(x)}\right]_{y \neq x}$ with concrete score matching:

$$\mathcal{L}_{\text{CSM}} = \frac{1}{2}\mathbb{E}_{x \sim p_t}\left[\sum_{y \neq x}\left(s_\theta(x_t,t)_y - \frac{p_t(y)}{p_t(x)}\right)^2\right] \quad (4)$$

Carnegie Mellon University

Meng et al. Concrete Score Matching: Generalized Score Matching for Discrete Data. NeurIPS 2022. https://arxiv.org/pdf/2211.00802

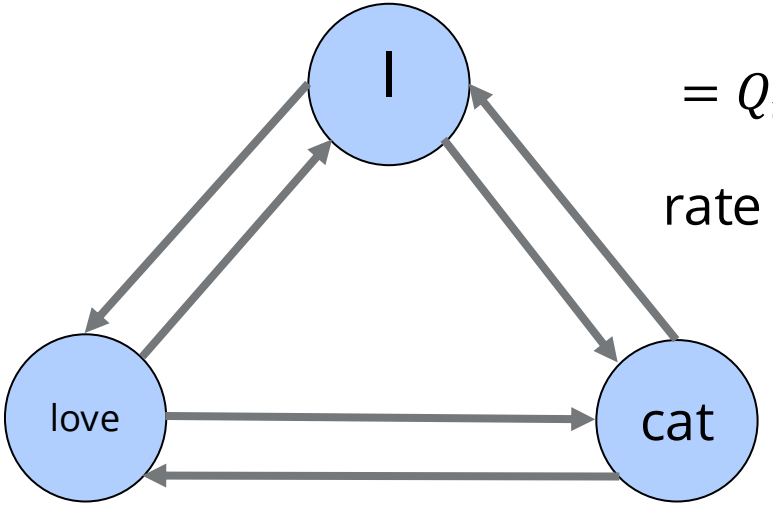
Problems with naïve concrete score matching

- The probability ratio should be **always positive** but CSM does not enforce that => -0.5 may look ok in loss function but it's actually super wrong
- Ratio explodes when p(x) is small
- MSE doesn't do well for relative error

Concrete Score Matching. Meng et al. (2022) generalizes the standard Fisher divergence in score matching, learning $s_\theta(x,t) \approx \left[\frac{p_t(y)}{p_t(x)}\right]_{y \neq x}$ with concrete score matching:

$$\mathcal{L}_{\text{CSM}} = \frac{1}{2}\mathbb{E}_{x \sim p_t}\left[\sum_{y \neq x}\left(s_\theta(x_t,t)_y - \frac{p_t(y)}{p_t(x)}\right)^2\right] \quad (4)$$

Let's derive a better loss



$$\bar{Q}_t(\text{“}I\text{”},\text{”}cat\text{”}) = Q_t(\text{“}cat\text{”},\text{“}I\text{”})\frac{p_t(\text{“}cat\text{”})}{p_t(\text{“}I\text{”})}$$

rate = base rate*ratio

Can be represented by a Poisson

Let's derive a better loss

Let $p_{\text{data}}(k) = \text{Poisson}(k; r) = e^{-r}\frac{r^k}{k!}$, $p_\theta(k) = \text{Poisson}(k; s_\theta) = e^{-s_\theta}\frac{{s_\theta}^k}{k!}$

Then the KL between the two distribution is

$$D_{\text{KL}}(p_{\text{data}}||p_\theta) = \sum_k p_{\text{data}}(k)\log\frac{p_{\text{data}}(k)}{p_\theta(k)} = \sum_k p_{\text{data}}(k)\log\frac{e^{-r}\frac{r^k}{k!}}{e^{-s_\theta}\frac{{s_\theta}^k}{k!}}$$

$$= \sum_k p_{\text{data}}(k)\,(-r + s_\theta) + k(\log r - \log s_\theta)$$

$$= (-r + s_\theta)\sum_k p_{\text{data}}(k) + (\log r - \log s_\theta)\sum_k p_{\text{data}}(k)\,k$$

$$= (-r + s_\theta) + (\log r - \log s_\theta)r$$

$$= s_\theta - r\log s_\theta$$

Score entropy loss

$$\mathbb{E}_{x\sim p_{\text{data}}}\left[\sum_{y\sim x} s_\theta(x)_y - \frac{p_{\text{data}}(y)}{p_{\text{data}}(x)}\log s_\theta(x)_y\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{y\sim x_i} s_\theta(x_i)_y - \frac{p_{\text{data}}(y)}{p_{\text{data}}(x_i)}\log s_\theta(x_i)_y\right]$$

- Log form -> everything is always positive
- Deals with crazy ratios better
- Distribution divergence rather than absolute error

Carnegie Mellon University

Lou et al. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. ICML 2024. https://arxiv.org/pdf/2310.16834

We can also apply the reparameterization trick here

Because $p(x_t) = \sum_{x_0} p(x_t|x_0)p_0(x_0)$ we can have

Theorem 3.4 (Denoising Score Entropy). *Suppose p is a perturbation of a base density p_0 by a transition kernel $p(\cdot|\cdot)$, ie $p(x) = \sum_{x_0} p(x|x_0)p_0(x_0)$. The score entropy $\mathcal{L}_{\text{SE}}$ is equivalent (up to a constant independent of θ) to the* ***denoising score entropy*** *$\mathcal{L}_{\text{DSE}}$ is*

$$\mathop{\mathbb{E}}_{\substack{x_0 \sim p_0 \\ x \sim p(\cdot|x_0)}} \left[\sum_{y \neq x} w_{xy} \left(s_\theta(x)_y - \frac{p(y|x_0)}{p(x|x_0)} \log s_\theta(x)_y \right) \right] \tag{7}$$

Now you have score entropy discrete diffusion (SEDD)!

Let’s look at them one by one

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise 

- Score-based models

  => Learning the score function 

- Flow matching

  => Learning the velocity

Discrete Diffusion:

- Discrete denoising diffusion models

  => Categorical noise

- Score entropy discrete diffusion

  => Concrete score & score entropy

- ?

Notice how right now all our models are for generic transition matrix

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise 

- Score-based models

  => Learning the score function 

- Flow matching

  => Learning the velocity

Discrete Diffusion:

- Discrete denoising diffusion models

  => Categorical noise

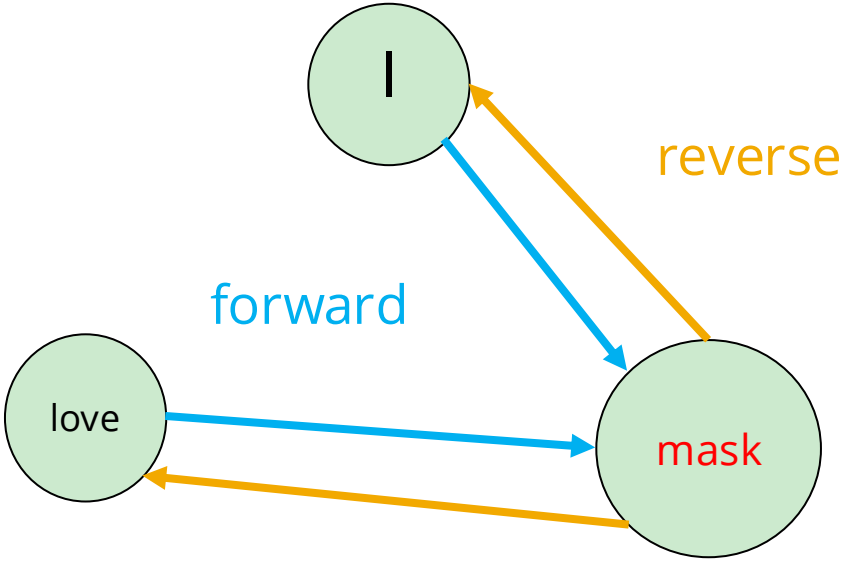
- Score entropy discrete diffusion

  => Concrete score & score entropy

- ?

What would be the easiest transition matrix to have

How about we can only mask/unmask tokens



Then the transition matrix is also simple

$$Q_t = \begin{bmatrix} 1-\beta_t & 0 & \beta_t \\ 0 & 1-\beta_t & \beta_t \\ 0 & 0 & 1 \end{bmatrix}$$

Transition rate to stay unmasked

Transition rate to <mask>

Actually, it can even be simpler

Because we are essentially only trying to interpolate between a clean data sample and the full mask, we can literally formulate our forward process like so

$$q_t(x_t|x_0) = \text{Cat}(\alpha_t x_0 + (1-\alpha_t)m)$$

Or we can write it as

$$q_t(x_t|x_0) = \begin{cases} \alpha_t, & \text{if } x_t = x_0 \\ 1-\alpha_t, & \text{if } x_t = m \\ 0, & \text{otherwise} \end{cases}$$

And $q_t(x_t = x_0|x_{t-1} = x_0) = \frac{\alpha_t}{\alpha_{t-1}}, q_t(x_t = m|x_{t-1} = x_0) = 1 - \frac{\alpha_t}{\alpha_{t-1}}$

$$q_t(x_t = m|x_{t-1} = m) = 1$$

Carnegie Mellon University

Sahoo et al. Simple and Effective Masked Diffusion Language Models. NeurIPS 2024.

Actually, it can even be simpler

What's even nicer is that now the super complicated $q_t(x_{t-1}|x_t, x_0)$ also becomes easy

- If $x_t = x_0$, $x_{t-1} = x_0$ deterministically (because we can't unmask after masking)
- If $x_t = m$, then $q_t(x_{t-1} = x_0|x_t = m, x_0) = \frac{q(x_t = m|x_{t-1} = x_0)q(x_{t-1} = x_0|x_0)}{q(x_t=m|x_0)}$

$$= \frac{(1 - \alpha_t/\alpha_{t-1})\alpha_{t-1}}{1 - \alpha_t} = \frac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t}$$

$$q_t(x_{t-1} = m|x_t = m, x_0) = 1 - \frac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t} = \frac{1 - \alpha_{t-1}}{1 - \alpha_t}$$

Carnegie Mellon University

Actually, it can even be simpler

$$q_t(x_{t-1}|x_t, x_0) = \begin{cases} \text{Cat}(x_0), & \text{if } x_t = x_0 \\ \frac{1-\alpha_{t-1}}{1-\alpha_t}, & \text{if } x_t = m, x_{t-1} = m \\ \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}, & \text{if } x_t = m, x_{t-1} = x_0 \end{cases}$$

Again, because $p_\theta(x_{t-1}|x_t) \approx \sum_{x_0} q(x_{t-1}, x_t|x_0)\, p_\theta(x_0|x_t)$

$$p_\theta(x_{t-1}|x_t = m) = \begin{cases} \frac{1-\alpha_{t-1}}{1-\alpha_t}, & \text{if } x_t = m, x_{t-1} = m \\ \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t} p_\theta(x_0|x_t), & \text{if } x_t = m, x_{t-1} = x_0 \end{cases}$$

Only need to predict the logits of the final clean output

The objective function is also super simple

Continuous limit of $\frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}$

$$\mathcal{L}_{\text{NELBO}}^{\infty} = \mathbb{E}_q \int_{t=0}^{t=1} \frac{\alpha_t'}{1-\alpha_t} \sum_{\ell=1}^{L} \log \langle \mathbf{x}_\theta^\ell(\mathbf{z}_t), \mathbf{x}^\ell \rangle dt$$

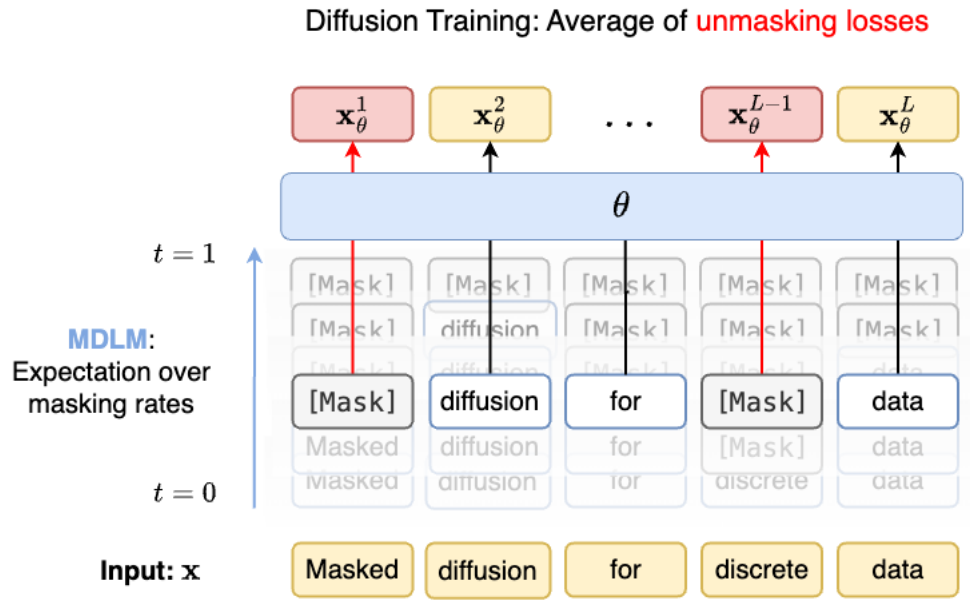
Cross entropy between the predicted clean sample and the data

Now you have masked diffusion language model!

Carnegie Mellon University

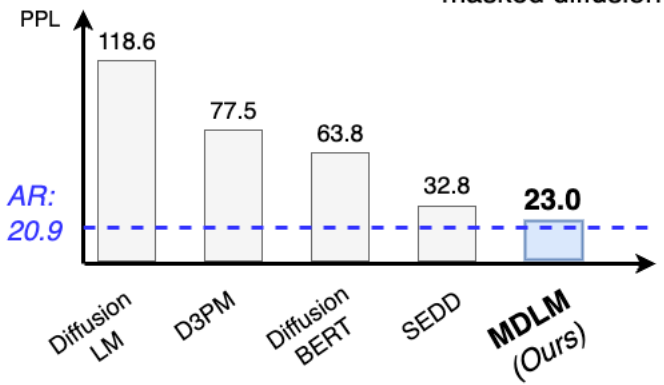
Sahoo et al. Simple and Effective Masked Diffusion Language Models. NeurIPS 2024. https://arxiv.org/pdf/2406.07524

MLDM works pretty well!



Simplified Masked Diffusion LM

- Masking rate is **random**, not fixed
- Objective is a **variational lower bound**
- Admits fast **ancestral sampling**
- Objective is a **simple average** of MLM losses
- ✓ Improved implementation relative to previous masked diffusion



Carnegie Mellon University

MLDM was also co-discovered by two concurrent works

- Sahoo et al. Simple and Effective Masked Diffusion Language Models. NeurIPS 2024
- Shit et al. Simplified and Generalized Masked Diffusion for Discrete Data. NeurIPS 2024

said

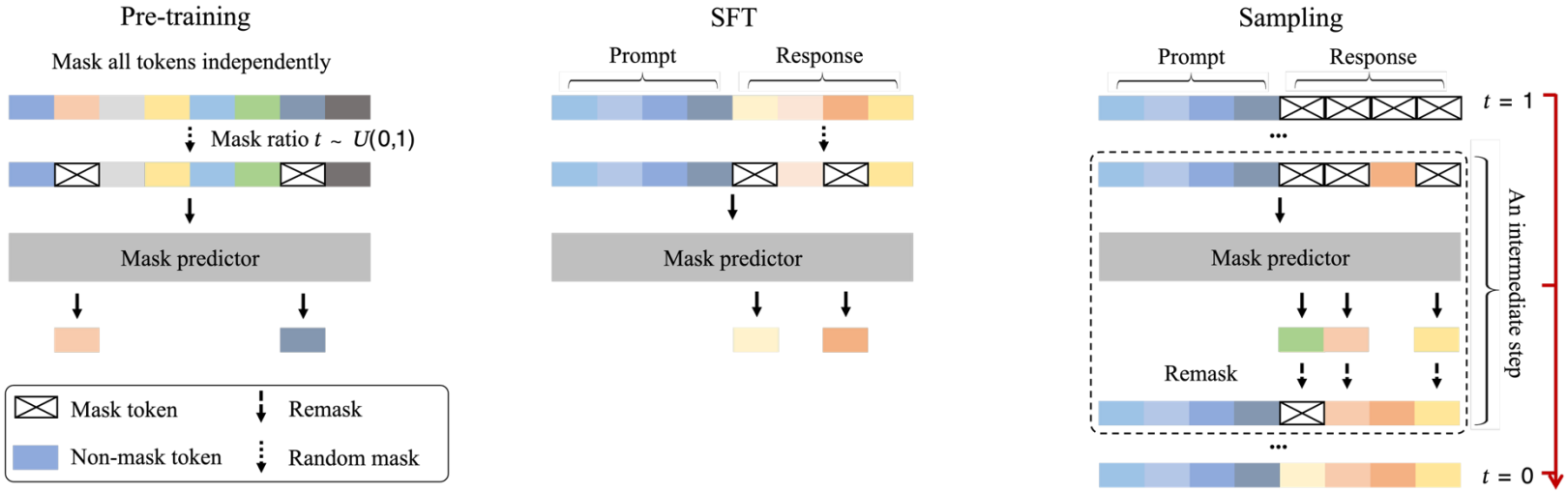
Carnegie Mellon University

Sahoo et al. Simple and Effective Masked Diffusion Language Models. NeurIPS 2024. https://arxiv.org/pdf/2406.07524

People have scaled it up!

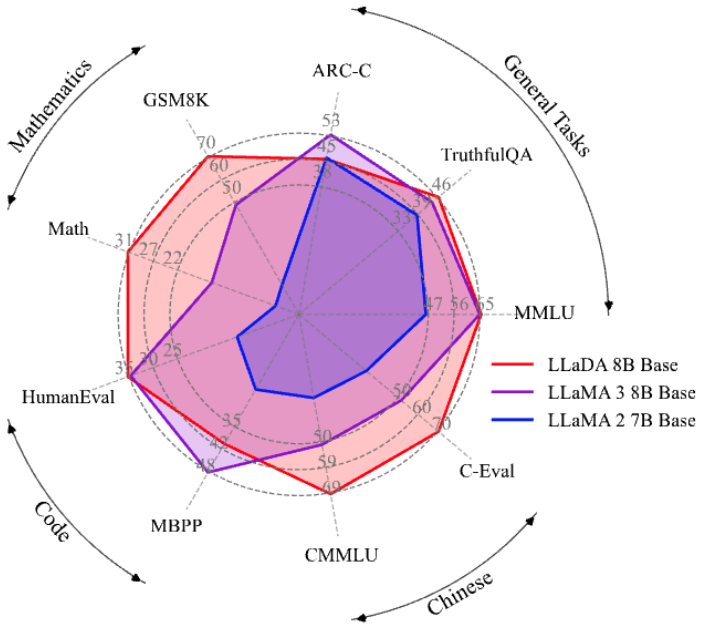
$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i=\mathrm{M}]\log p_\theta(x_0^i|x_t)\right],$$

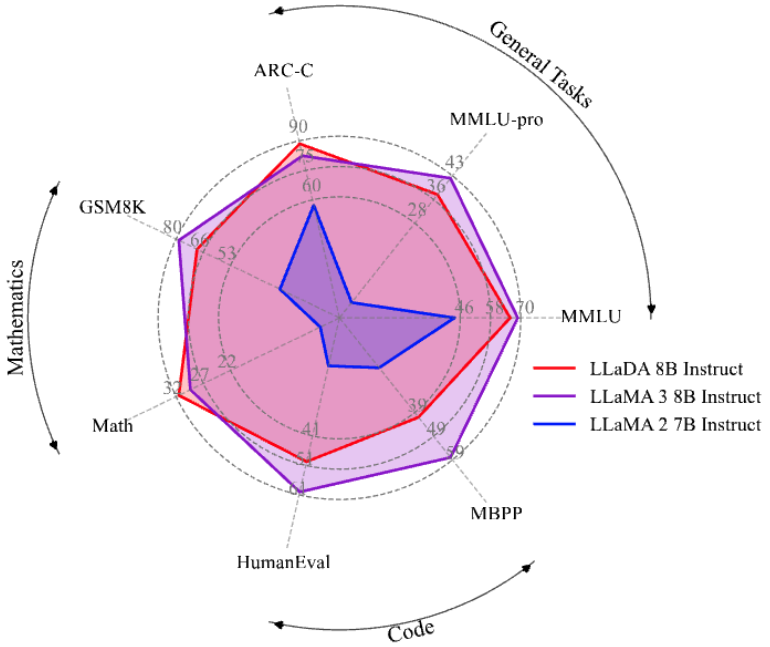
Large language diffusion with masking (LLaDA)



Nie et al. Large Language Diffusion Models. NeurIPS 2025. https://arxiv.org/pdf/2502.09992

LLaDA works comparably to LLaMA!





Carnegie Mellon University

Let’s look at them one by one

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise 

- Score-based models

  => Learning the score function 

- Flow matching

  => Learning the velocity 

Discrete Diffusion:

- Discrete denoising diffusion models

  => Categorical noise

- Score entropy discrete diffusion

  => Concrete score & score entropy

- ?
- Relationships/Interpolation with LLM?

Carnegie Mellon University