Carnegie Mellon University

Lecture 13: Discrete Flow Matching & Edit Flow

Yutong (Kelly) He

10-799 Diffusion & Flow Matching, Feb 19th, 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
- There shall be some survey this weekend/next week for some extra credits

Carnegie Mellon University

So far we have learned

Continuous Diffusion:

- Denoising diffusion models

  => Adding noise and learning to denoise ✅

- Score-based models

  => Learning the score function ✅

- Flow matching

Discrete Diffusion:

- Discrete denoising diffusion models

  => Categorical noise

- Score entropy discrete diffusion

  => Concrete score & score entropy

How to add noise in text

Andrej Karpathy
@karpathy

Copy Thread

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.

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 errorəmessages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual con🧧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], love: [0,1,0,], cat: [0,0,1]

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:

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

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

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

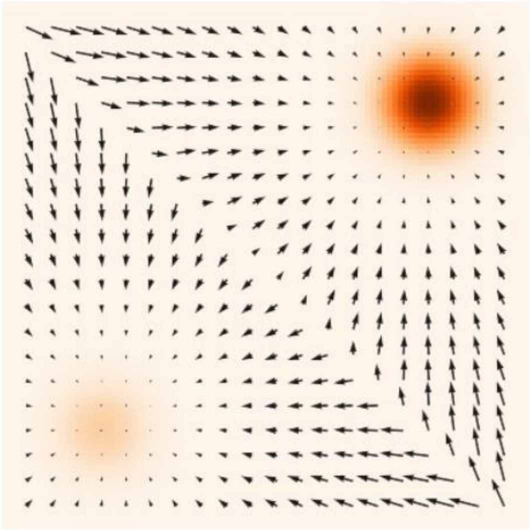
Where $\bar{Q}_t = Q_1 Q_2 \dots Q_t$

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

Carnegie Mellon University

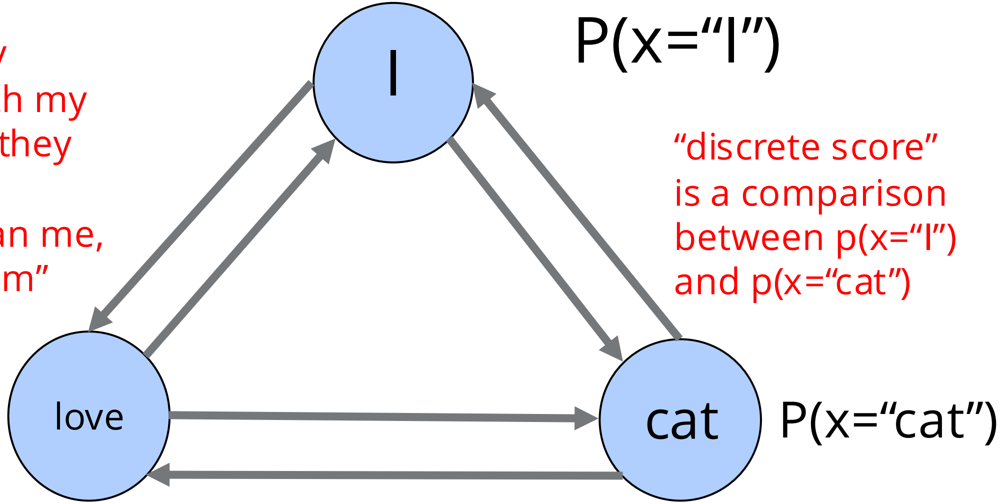
The continuous score v.s. the "discrete score"

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*](https://arxiv.org/pdf/2211.00802)

Score entropy discrete diffusion

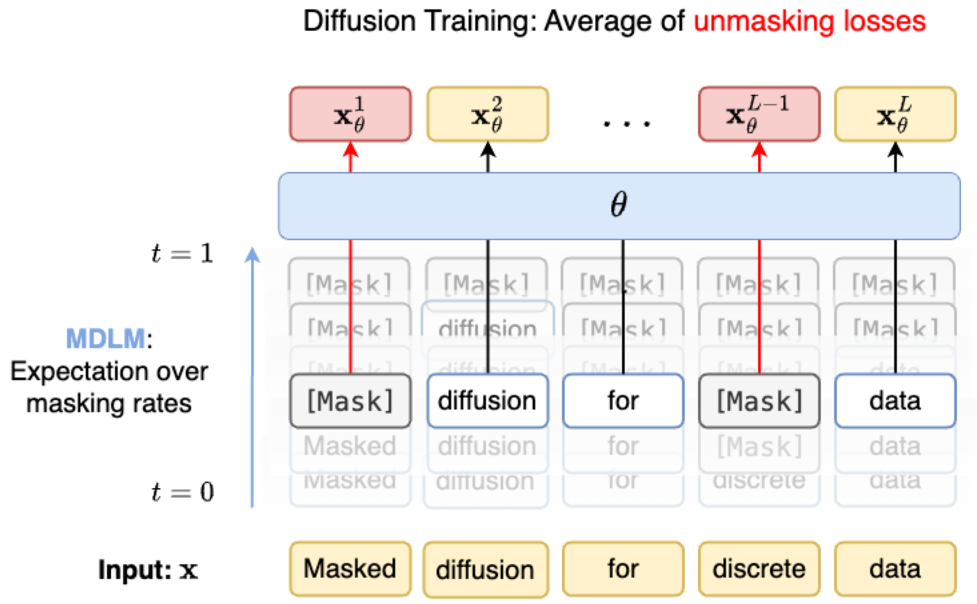
Because $p(x_t) = \sum_{x_0} p(x_t|x_0)p_0(x_0)$ we can have

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 $\mathcal{L}_{\mathrm{SE}}$ is equivalent (up to a constant independent of θ) to the **denoising score entropy** $\mathcal{L}_{\mathrm{DSE}}$ is*

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

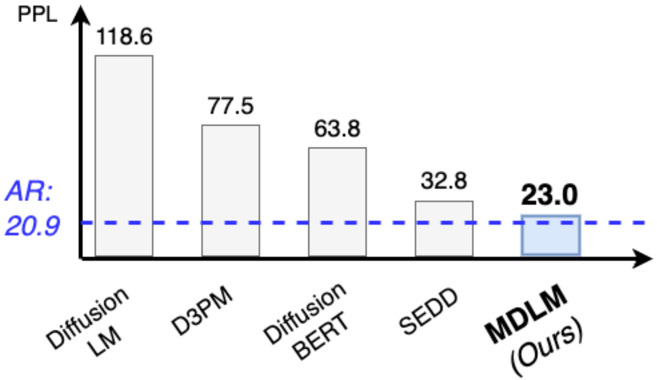
Carnegie Mellon University

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

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

So far we have learned

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 is the analogy to velocity in discrete setting?

Let’s recall what we need to build a flow model

In order for a CNF to model transports between probability distributions, we need the following assumptions:

- **Conservation of mass:** No new mass and mass does not disappear

  => Probability always adds up to 1

- **Continuity equation:** Not only that the mass is conserved, it also does not teleport

  => Probability can only move/change continuously

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

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

This means that $\sum_z p_{t+dt}(z|x_t) = \sum_z I_{x,z} + Q_{x,z}dt = 1 + \sum_z Q_{x,z}dt = 1$, therefore

$$\sum_z Q_t(x,z) = 0$$

Conservation of mass

$$Q_t(x,x) = -\sum_{z\neq x} Q_t(x,z)$$

Carnegie Mellon University

Continuity equation for CTMC (a.k.a Kolmogorov equation)

Now all we need is to deduce how the probability changes over time, which can be written as

$$\frac{dp_t}{dt} = \underbrace{\sum_{z \neq x} Q_t(z,x)\, p_t(z)}_{\text{Incoming}} - \underbrace{\sum_{z \neq x} Q_t(x,z)\, p_t(x)}_{\text{Outgoing}}$$

Because $Q_t(x,x) = -\sum_{z \neq x} Q_t(x,z)$, this can also be written as

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

We just need to simulate this ODE!

Carnegie Mellon University

Simulating a discrete probability path trajectory

With $\frac{dp_t}{dt} = Q_t p_t$, we can have

$$p_{t+\Delta t}(x_{t+\Delta t}) = p_t(x_t) + \Delta t \frac{dp_t}{dt}$$

$$= p_t(x_t) + \Delta t \sum_{x_t} Q_t(x_t, x_{t+\Delta t}) p_t(x_t)$$

Our “velocity” a.k.a the

sampling rate matrix

Parametrizing & training the rate matrix

We can yet again parameterize the rate matrix w.r.t. the clean data estimation

$$Q_t(x_t, z) = \sum_{x_1} Q_t(x_t, z|x_1) p_1(x_1|x_t)$$

The only thing we need to learn

Can directly get closed form solution via the forward process

So we yet again, train a clean data predictor $p_\theta(x_1|x_t)$ via cross entropy

Now you have discrete flow matching!!

Carnegie Mellon University

Discrete flow matching sampling

Start from $t = 0$ and some random initial sample x_0,

While $t < 1$:

$Q_\theta(x_t, z, t) = \sum_{x_1} Q_t(x_t, z|x_1) p_\theta(x_1|x_t)$ for all z

Sample $x_{t+\Delta t}$ from Categorical distribution $\mathrm{Cat}(1(x_t == x_{t+\Delta t}) + Q_\theta(x_t, x_{t+\Delta t}, t)\Delta t)$

$t = t + \Delta t$

Carnegie Mellon University

Campell et al. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. ICML 2024. https://arxiv.org/pdf/2402.04997

Two discrete flow matching papers

- Campell et al. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. ICML 2024. https://arxiv.org/pdf/2402.04997
- Gat et al. Discrete Flow Matching. NeurIPS 2024. https://arxiv.org/pdf/2407.15595

So far we have learned

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

- Discrete flow matching

  => Learning the rate matrix

Carnegie Mellon University

Are discrete diffusion models perfect?

Things we can improve discrete diffusion on

- The model cannot generate variable length paragraphs – it has to be fixed length!
- The model has to use bidirectional attention – no KV caching!
- Better ways to decide the unmasking order/how many tokens to unmask
- Continuous + discrete (a.k.a multimodal) diffusion?

How would you generate a variable-length paragraph

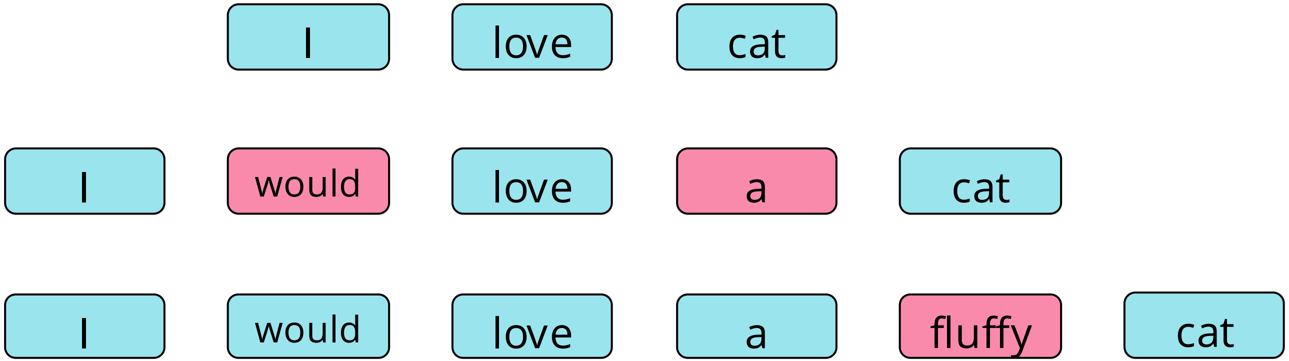
- You can use an LLM and generate everything autoregressively, i.e. insert one token at the end of the sentence at a time

How would you generate a variable-length paragraph

- You can use an LLM and generate everything autoregressively, i.e. **insert** one token at the end of the sentence at a time

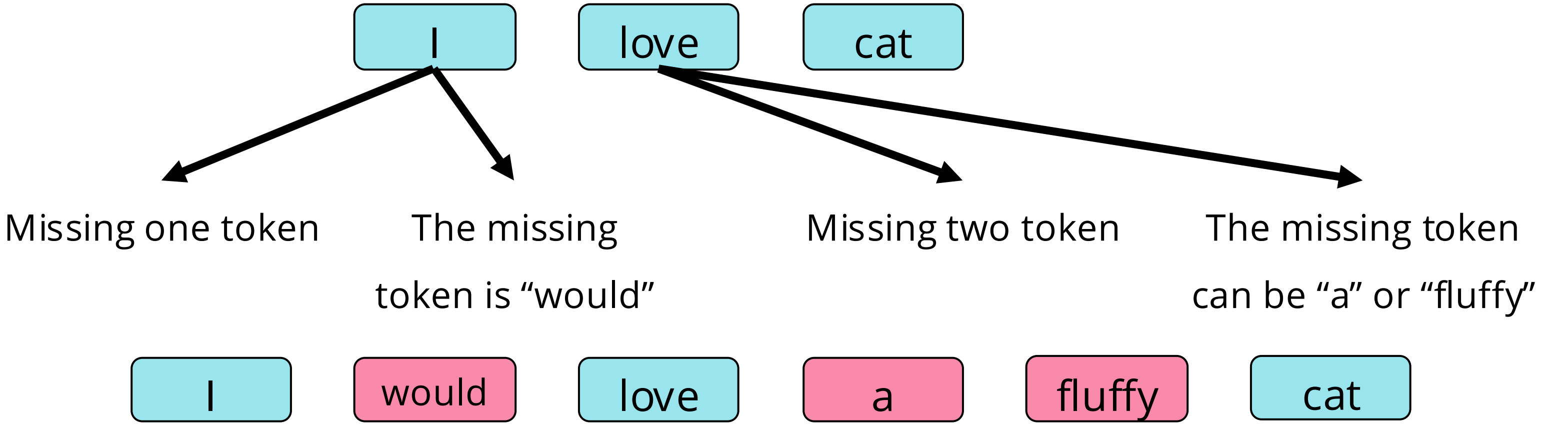
Why do we have to insert only at the end? Why can't we also insert at other places in the sentence?

Insertion-based generation

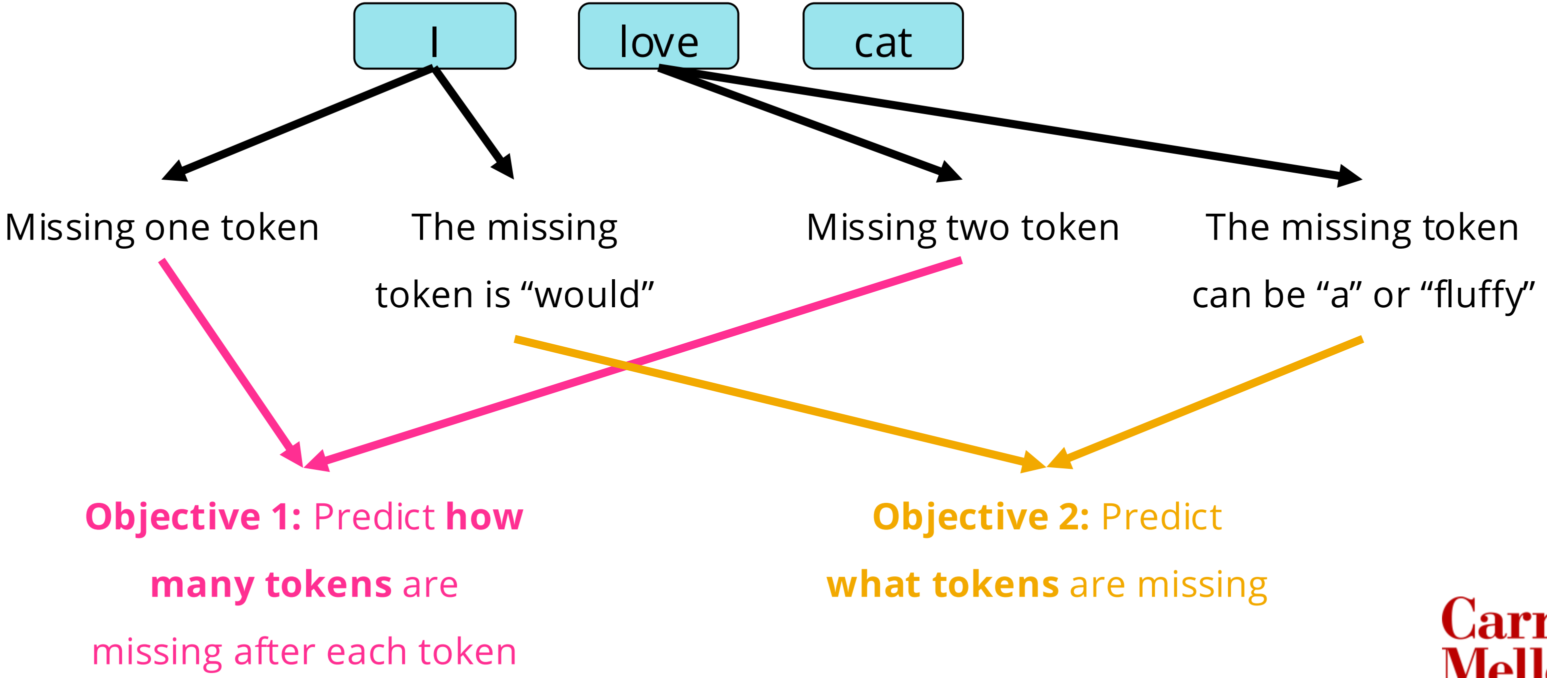


Nguyen et al. OneFlow: Concurrent Mixed-Modal and Interleaved Generation with Edit Flows. https://arxiv.org/pdf/2510.03506

How to model insertion-based generation



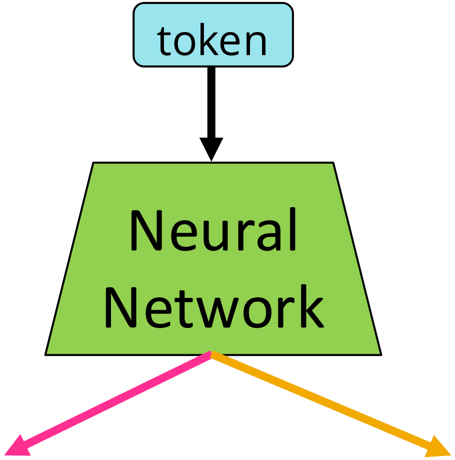
How to model insertion-based generation



Carnegie Mellon University

Nguyen et al. OneFlow: Concurrent Mixed-Modal and Interleaved Generation with Edit Flows. https://arxiv.org/pdf/2510.03506

How to model insertion-based generation



how many tokens are missing after this token

One possible token that is missing

KL between two Poissons

Cross entropy of all tokens missing from the ground truth

Carnegie Mellon University

Nguyen et al. OneFlow: Concurrent Mixed-Modal and Interleaved Generation with Edit Flows. https://arxiv.org/pdf/2510.03506

You can also add other edit operations

	I	love	cat		
I	would	love	a	cat	
I	would	love	a	fluffy	cat
I	would	love	~~a~~	fluffy	dogs

Now you have Edit Flow!

Havasi et al. Edit Flows: Flow Matching with Edit Operations. NeurIPS 2025. https://arxiv.org/pdf/2506.09018

How to model Edit Flow

Remember how in masked diffusion model we got something like this

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

$= \frac{\alpha_t'}{1-\alpha_t}$ at limit

When something happens

We can basically parametrize the edits as the rate $\lambda_\theta(x_t, t) \approx \frac{\alpha_t'}{1-\alpha_t}$ and the logit $p_\theta(x_1|x_t) \approx p_1(x_1|x_t)$

Rates can be trained with KL between Poissons and logits can be trained with cross entropy

(Deletion only has rate and doesn't need logits)

Carnegie Mellon University

Havasi et al. Edit Flows: Flow Matching with Edit Operations. NeurIPS 2025. https://arxiv.org/pdf/2506.09018

Things we can improve discrete diffusion on

- The model cannot generate variable length paragraphs – it has to be fixed length! ✅
- The model has to use bidirectional attention – no KV caching!
- Better ways to decide the unmasking order/how many tokens to unmask
- Continuous + discrete (a.k.a multimodal) diffusion?

Block diffusion/Semi-autoregressive MDLM

Autoregression: ✓ High quality ✓ Arbitrary-length ✓ KV caching ✗ Not Parallelizable

Generation steps

```
There are three categories of the average
There are three categories of the average rate
There are three categories of the average rate of...
```

Diffusion: ✗ Lower quality ✗ Fixed-length ✗ No KV caching ✓ Parallelizable

```
      the reusability                        will continue to       the
Repeal the reusability cuts and        the law will continue to reduce the
Repeal the reusability cuts and prove the law will continue to reduce the deficit.
```

Block Diffusion (Ours): ✓ High quality ✓ Arbitrary-length ✓ KV caching ✓ Parallelizable

```
On September 17,       we      be
On September 17, 2016, we will be giving the      release of
On September 17, 2016, we will be giving the beta-release of the      to our server      testing ...
```

Arriola et al. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. https://arxiv.org/pdf/2503.09573

Block diffusion/Semi-autoregressive MDLM

LLM: ✅ High quality ✅ Arbitrary-length ✅ KV caching ❌ Not parallelizable

`There`

Masked Diffusion: ❌ Lower quality ❌ Fixed-length ❌ No KV caching ✅ Parallelizable

Block Diffusion: ✅ High quality ✅ Arbitrary-length ✅ KV caching ✅ Parallelizable

Carnegie Mellon University

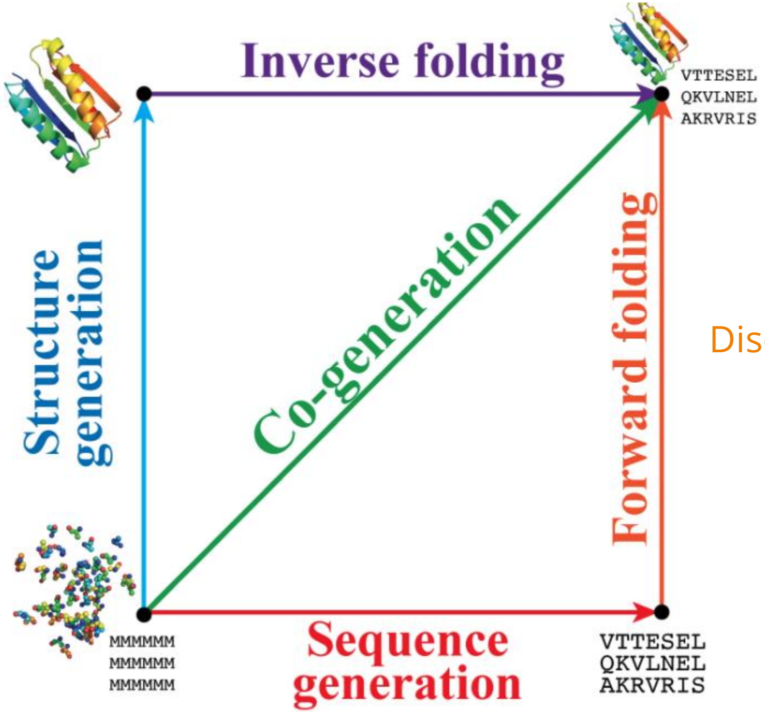
Arriola et al. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. https://arxiv.org/pdf/2503.09573

Things we can improve discrete diffusion on

- The model cannot generate variable length paragraphs – it has to be fixed length! ✅
- The model has to use bidirectional attention – no KV caching! ✅
- Better ways to decide the unmasking order/how many tokens to unmask ✅
- Continuous + discrete (a.k.a multimodal) diffusion?

Multimodal flow

Flow matching

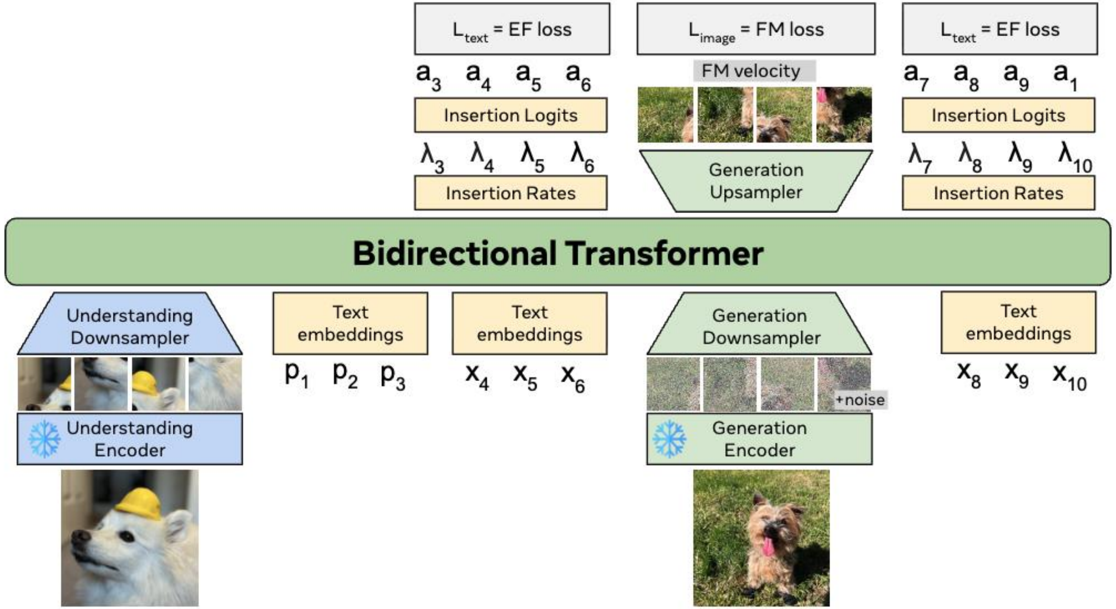


Discrete flow matching

Carnegie Mellon University

Campell et al. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. ICML 2024. https://arxiv.org/pdf/2402.04997

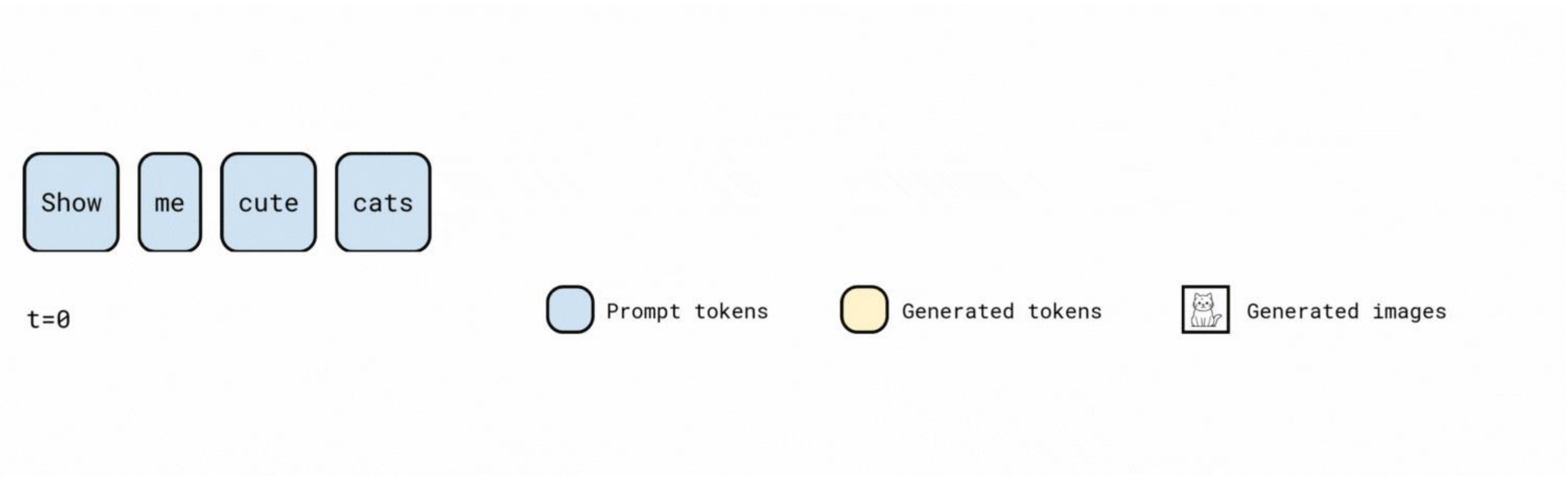
OneFlow



Carnegie Mellon University

Nguyen et al. OneFlow: Concurrent Mixed-Modal and Interleaved Generation with Edit Flows. https://arxiv.org/pdf/2510.03506

OneFlow


Show
me
cute
cats
t=0
Prompt tokens
Generated tokens
Generated images

Things we can improve discrete diffusion on

- The model cannot generate variable length paragraphs – it has to be fixed length! ✅
- The model has to use bidirectional attention – no KV caching! ✅
- Better ways to decide the unmasking order/how many tokens to unmask ✅
- Continuous + discrete (a.k.a multimodal) diffusion? ✅

Congratulations! You are now a diffusion expert 🎉

This was what we said we were going to learn in this class, and I hope everyone now has some grasp of all of these

In this class, we are going to learn about **diffusion & flow matching** – the technology behind the great AI ~~memes~~ advancements in image generation & beyond in recent years. In particular, we will learn

- The **intuitions and math** behind these algorithms
- How to **implement** basic diffusion & flow matching models in Python
- How to **train a good image generation model** using GPUs
- **Techniques** that **improves** upon the vanilla models
- How to conceptually extend these frameworks to **discrete data**
- How to **demonstrate your thoughts** and **your work**

Carnegie Mellon University

All the resources will remain available forever

- Lecture recordings
- Lecture slides
- Class website
- Homework
- Discord

Now it's your turn to take what you have learned into the real world

- Open source your homework
- Turn your homework into a real research project
    - Pro tips: Poster session is a great time to see if anyone else is doing what you are interested in, and you can combine your efforts into a larger, more research like project
- Use what you have learned in this class in your own research/work in the future

Last chance for questions

Final reminder

- 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
- There shall be some survey this weekend/next week for some extra credits

Thanks everyone for joining me in this class! Now go explore the real world!

Image generated by Nano Banana