Diffusion Models

By Tejomay Kishor Padole
(PhD Student, IIT Bombay)

Generative Modeling

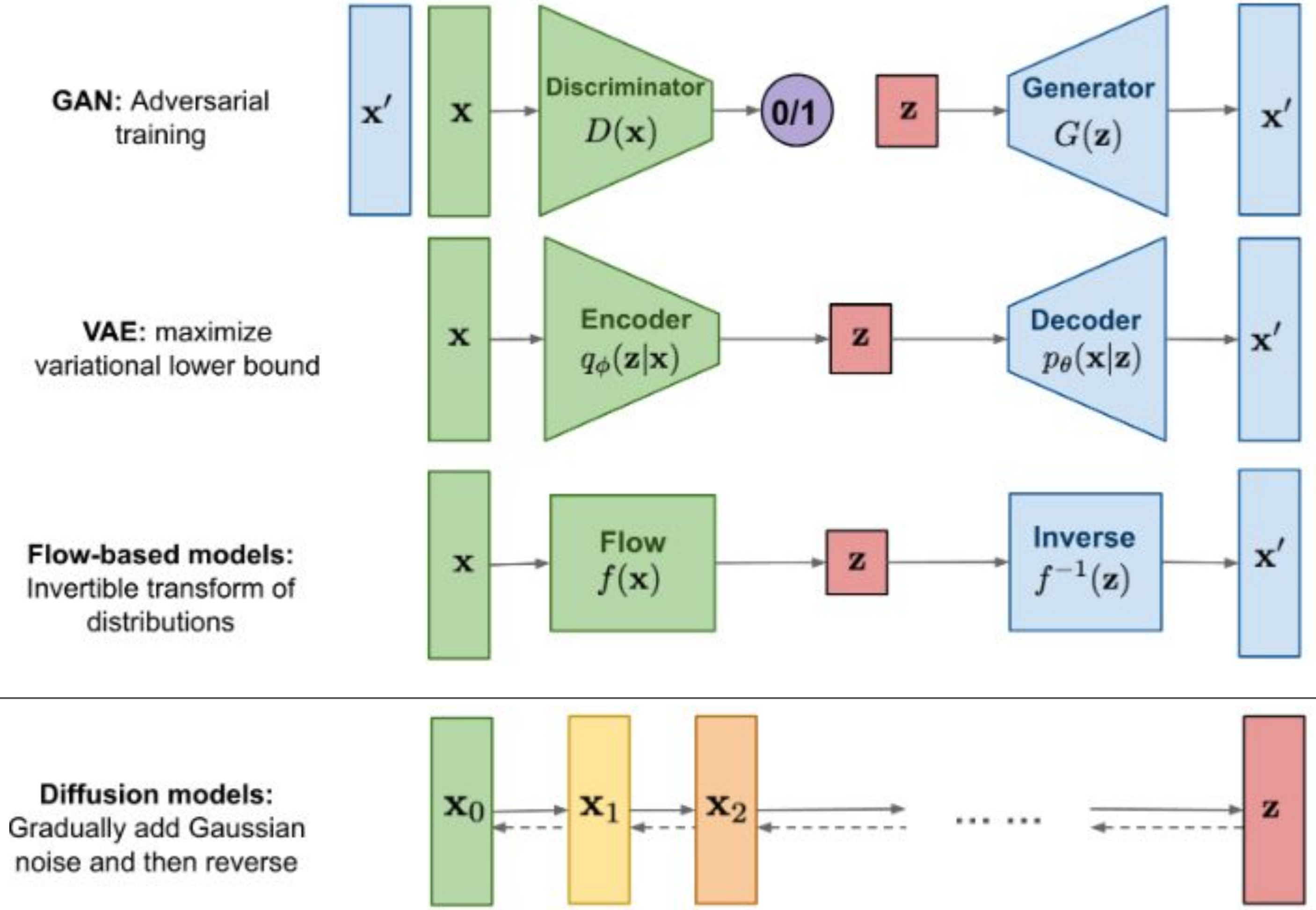
- Learning to generate new samples given a set of data points.

Estimate: $p(x_i) \;\; \forall x_i \in \mathcal{D}$

Generate: $x_{\text{new}} \sim p(x_i)$

Autoregressive LLMs, Variational Autoencoders, Normalizing flows	**Generative Adversarial Networks**
Explicitly model the data distribution	Modeling the generation process

Generative Models



Good quality samples, but cannot estimate the distribution

Can estimate the distribution, but lacked quality

Diffusion models did both!

Density estimation, high quality samples and better diversity!

[Lilian Weng's blog on Diffusion Models]

Why Diffusion Models are Trending?

- Diffusion models enabled high resolution image generation!

[SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis (2023)]

Diffusion Models

- Inspired from non-equilibrium statistical physics.

- **Forward process:** slowly and iteratively destroys the structure of the input until it gets completely transformed to random noise.

- **Reverse process:** learns to iteratively construct the data from random noise using a Neural Network.

- The noising process can be seen as similar to a drop of ink diffusing into clear water.

Image from https://chemistryclinic.co.uk/1-3-diffusion/

Generative Process as a form of Iterative Denoising

Noising Process

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

(Markovian Gaussian transitions)

$p_{data}(x)$

$\mathcal{N}(0, I)$

x_0 x_1 x_2 .... x_T

Scaling factor for variance preservation

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Variance of the forward process, monotonic increasing function of *t*

$$x_t = \underbrace{\sqrt{1-\beta_t}\, x_{t-1}}_{\text{Downscaled previous timestep image}} + \underbrace{\sqrt{\beta_t}\epsilon}_{\text{Noise component}} \quad \text{where } \epsilon \sim \mathcal{N}(0, I)$$

Picture taken from Stanford CS236 Lecture series.

Generative Process as a form of Iterative Denoising

Noising Process

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I$$

Direct Transition

$p_{data}(x)$

$\mathcal{N}(0, I)$

x_0 x_1 x_2 . . . . x_T

$$\text{We define } \alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

$$\boxed{q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)}$$

Part of the original image

Part of random gaussian noise

Picture taken from Stanford CS236 Lecture series.

Generative Process as a form of Iterative Denoising

Noising Process

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

(Markovian Gaussian transitions)

$p_{data}(x)$

$\mathcal{N}(0, I)$

x_0 x_1 x_2 .... x_T

$$p(x_T) = \mathcal{N}(x_T; 0, I) \quad \text{and}$$
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Parameterized with a Neural Network

How to learn this?

Picture taken from Stanford CS236 Lecture series.

Learning the Generative Process

$p_{data}(x)$  $\mathcal{N}(0, I)$

x_0 x_1 x_2 .... x_T

Generative Process

Initialize: $p(x_T) = \mathcal{N}(x_T; 0, I)$ and

Iteratively sample from: $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$

$$-\log p_\theta(x_0) \leq -\log p_\theta(x_0) + D_{\mathbf{KL}}(q(x_{1:T}|x_0) \,||\, p_\theta(x_{1:T}|x_0))$$

(Variational Lower Bound on log-likelihood of data)

$$= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T} \sim q(x_{1:T}|x_0)} \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)} \right]$$

Expanding KL-divergence

$$= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T} \sim q(x_{1:T}|x_0)} \left[\log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}} \right]$$

Re-writing the denominator

$$= \mathbb{E}_{x_{1:T} \sim q(x_{1:T}|x_0)} \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right]$$

Got rid of the log-likelihood term!

[Picture taken from Stanford CS236 Lecture series.]()

Learning the Generative Process

$p_{data}(x)$  $\mathcal{N}(0, I)$

x_0 x_1 x_2 .... x_T

Generative Process

$$L_{VLB} = \mathbb{E}_q\left[\cancel{\underbrace{D_{\mathbf{KL}}(q(x_T|x_0)\,||\,p_\theta(x_T))}_{L_T}} + \sum_{t=2}^{T}\underbrace{D_{\mathbf{KL}}(q(x_{t-1}|x_t, x_0)\,||\,p_\theta(x_{t-1}|x_t))}_{L_{t-1}} - \cancel{\underbrace{\log p_\theta(x_0|x_1)}_{L_0}}\right]$$

$$= \mathbb{E}_{x_0}\left[\frac{1}{2\,\|\Sigma_\theta\|_2^2}\|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|_2^2\right]$$

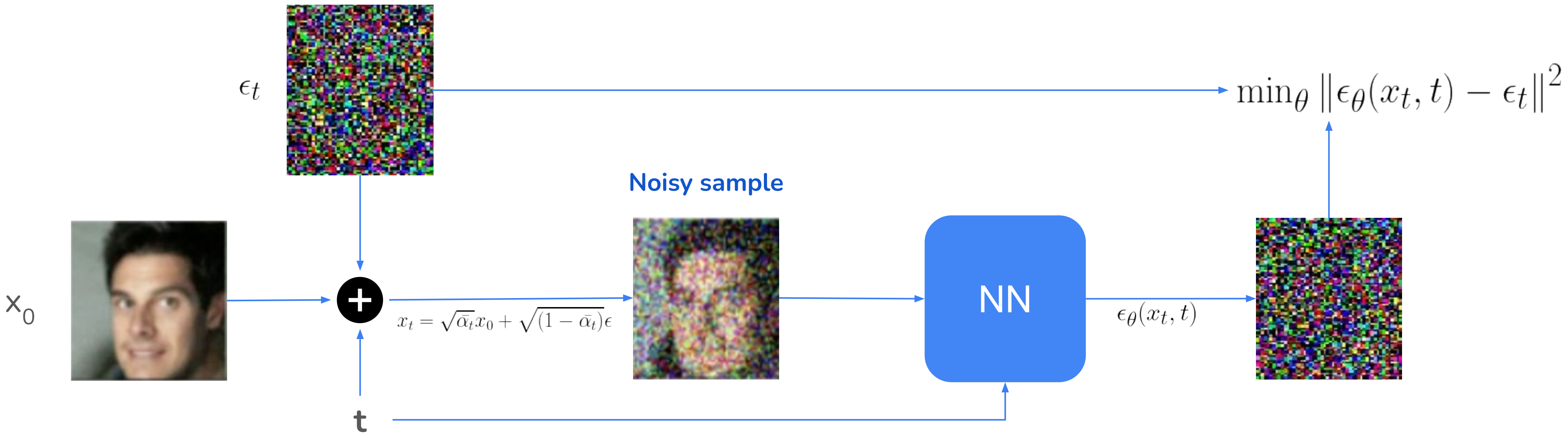
Closed-form solution for KLD between two gaussians (and variance ignored)

$$\underbrace{= \mathbb{E}_{x_t,\epsilon_t}\|\epsilon_t - \epsilon_\theta(x_t, t)\|^2}_{\text{Final Loss Function}}$$

Reparameterized as: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t))$

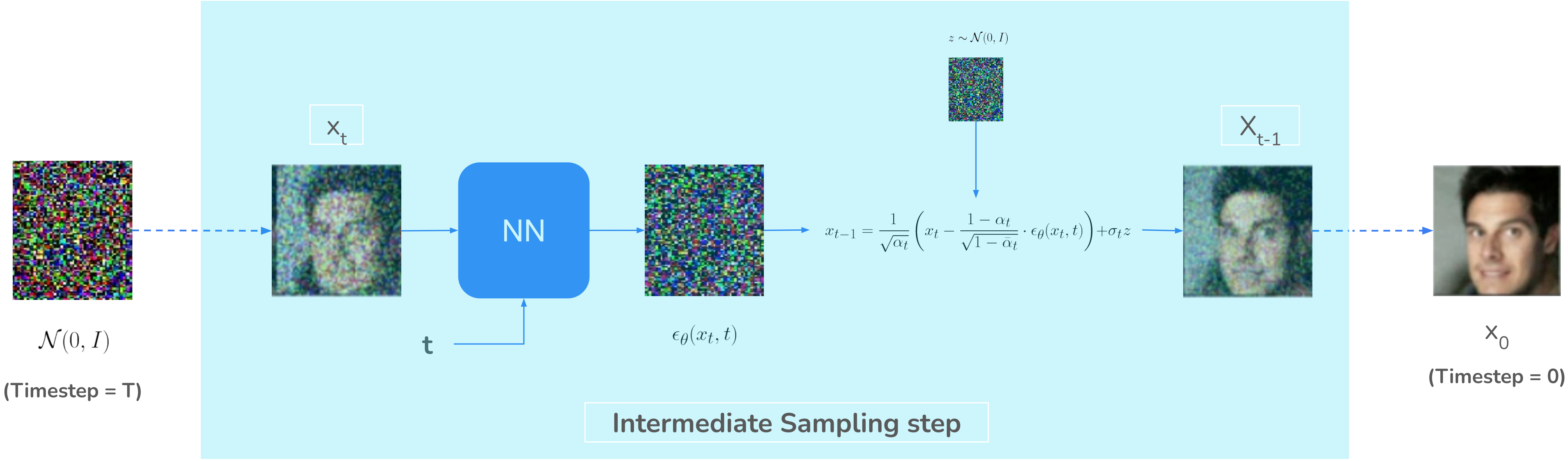
Picture taken from Stanford CS236 Lecture series.

Training Algorithm



[Denoising Diffusion Probabilistic Models (NeurIPS, 2020)]

Sampling (Generation) Algorithm



[Denoising Diffusion Probabilistic Models (NeurIPS, 2020)]

The Score Function and Diffusion Models

The Score Function

$$x_i \sim p(x)$$

Goal: Estimate the distribution with a NN

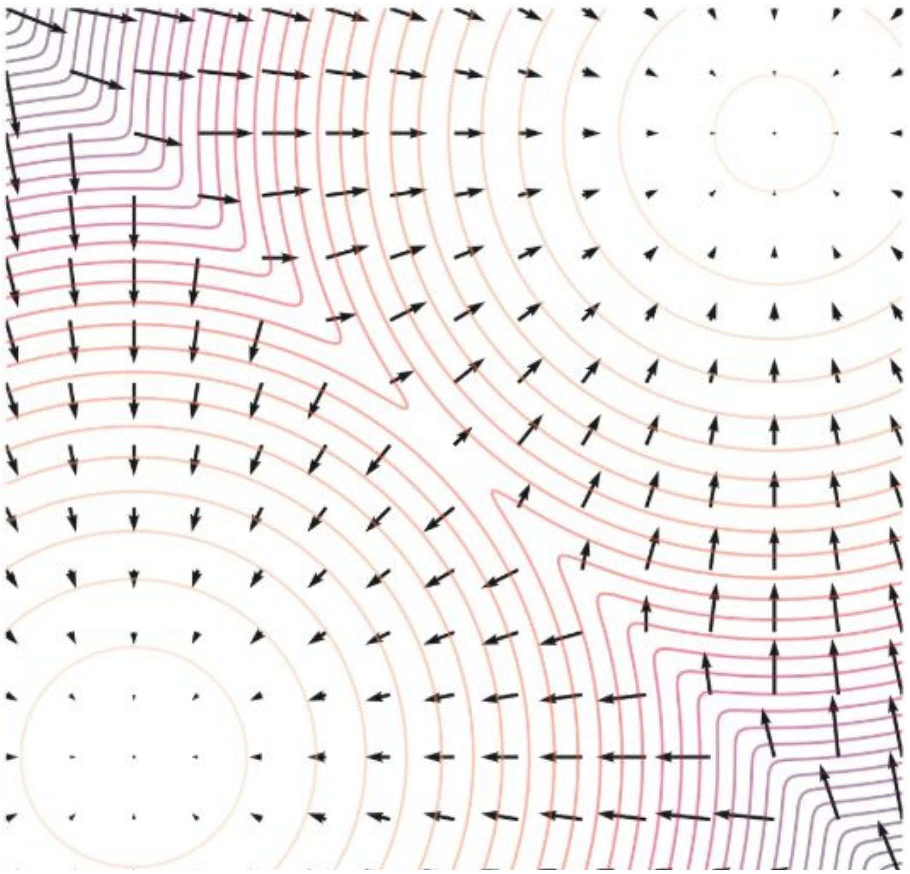
$$x_i \longrightarrow \boxed{\mathbf{NN}_\theta} \longrightarrow f_\theta(x_i)$$

$$p_\theta(x_i) = \frac{\exp\left(-f_\theta(x_i)\right)}{\boxed{Z_\theta}} \qquad \longrightarrow Z_\theta = \int_x \exp(-f_\theta(x))dx$$

Intractable!!

$$\ln p_\theta(x_i) = -f_\theta(x_i) - \ln Z_\theta$$

$$\underbrace{\nabla_{x_i} \ln p_\theta(x_i)}_{\textbf{Score function}} = -\nabla_{x_i} f_\theta(x_i) - \nabla_{x_i} \cancel{\ln} Z_\theta$$



Score function interpretation on a mixture of two gaussians in 2D.

Image from *Song et al., 2019*

Interpretation of the Score Function

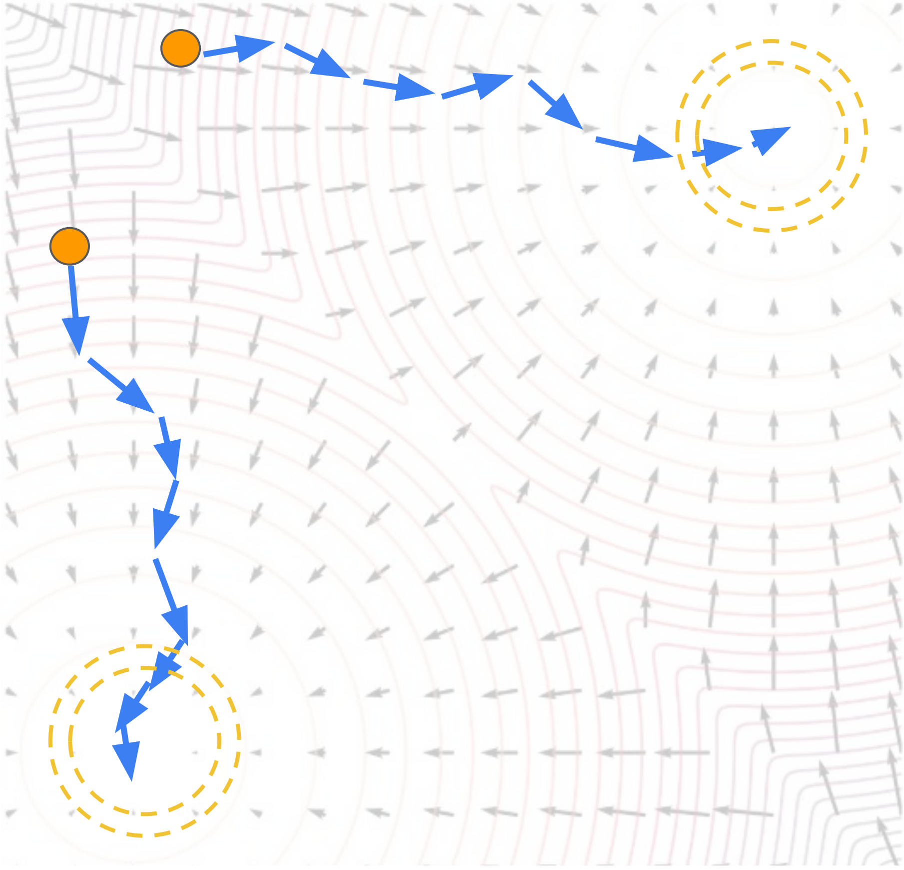
- Learning the score function allows sampling from the data distribution.

Step-size **Gaussian noise**

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad i = 0, 1, \cdots, K,$$

$$\mathbf{z}_i \sim \mathcal{N}(0, I)$$

Prevents mode collapse and allows for sampling from regions with low-data density



Langevin Dynamics

Generative Modeling by Estimating Gradients of the Data Distribution (Song et al., 2019)

Score-based Diffusion Models

Noising Process

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

$p_{data}(x)$

$\mathcal{N}(0, I)$

score function

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$$

Generative Process

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$$

(Time-conditioned score network)

Training objective: $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\arg\min}\, \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \Big[\big\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \big\|_2^2 \Big] \Big\}$

Score-based Generative Modeling through Stochastic Differential Equations (Song et al., 2021)

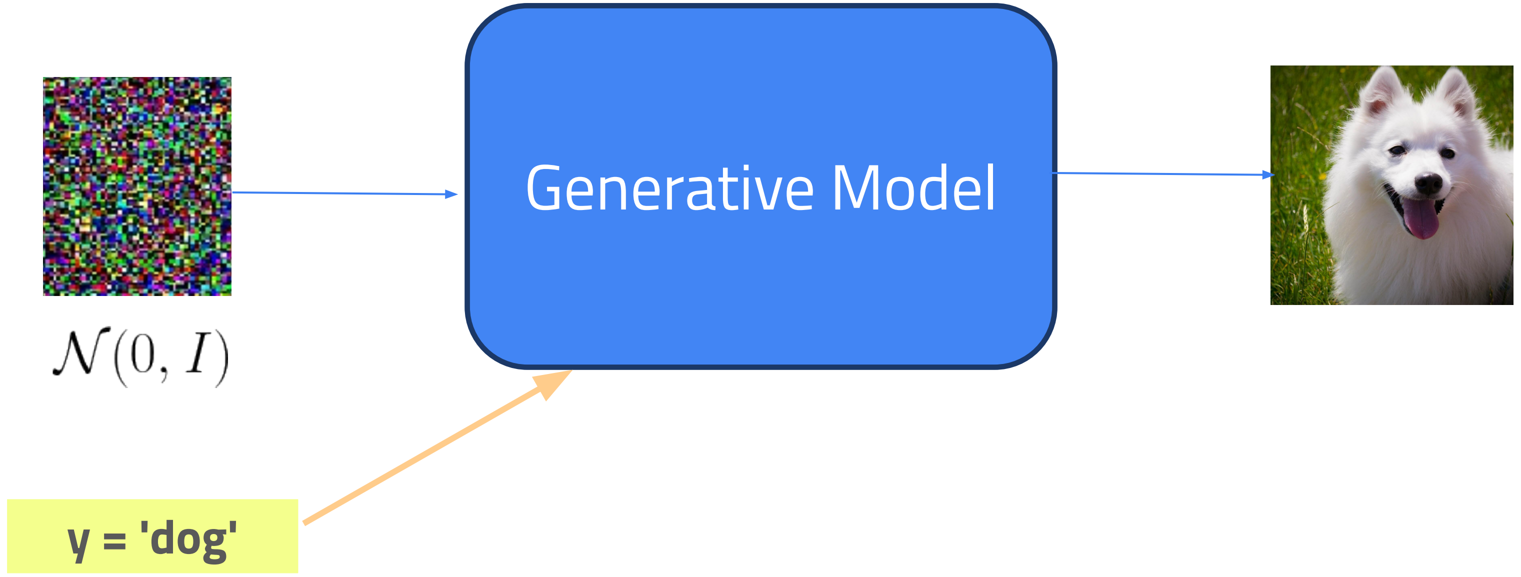
Conditioning of Diffusion Models

Class Conditional Diffusion Models

Generate: $x_{\text{new}} \sim p(x_i)$ (Unconditional)

What if we want to generate: $x_{new} \sim p(x_i|y)$ (where 'y' is a class label)

General strategy: Train the generative model with class awareness.



Classifier Guided Diffusion

We want: $x_{\text{new}} \sim p(x|y)$ (where 'y' is a class label)

$$p(x \mid y) = \frac{p(y \mid x) \cdot p(x)}{p(y)}$$

$$\log p(x \mid y) = \underbrace{\log p(y \mid x)}_{\text{Label probabilities}} + \underbrace{\log p(x)}_{\text{Unconditional data probability}} - \cancel{\log p(y)}$$

Label probabilities (can be obtained from a pre-trained image classifier

Unconditional data probability (modeled using a diffusion model)

Dhariwal and Nichol, 2021 **propose an additional guidance scale:**

$$\log p_\gamma(x|y) \propto \log p(x) + \boxed{\gamma} \log p(y|x)$$

Guidance scale

Diffusion models beat GANs on image synthesis (NeurIPS, 2021)

Classifier Guided Diffusion

- Varying the guidance scale allows us to have a trade-off between diversity and fidelity.

$$\boxed{\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x).}$$

γ = 1.0

γ = 10.0

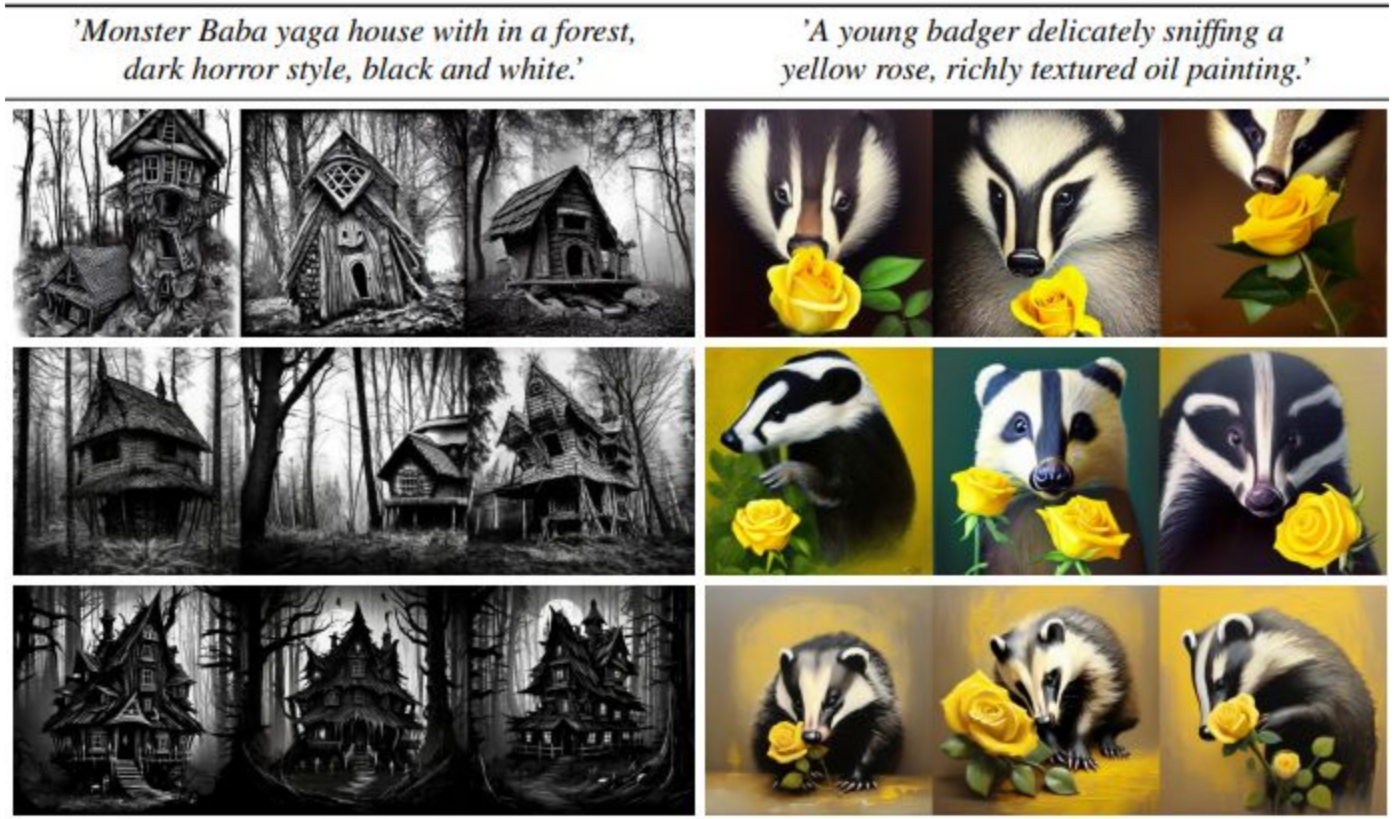
Generated images from the class **"Pembroke Welsh Corgi"** with different guidance scales.

Diffusion models beat GANs on image synthesis (NeurIPS, 2021)

Arbitrary Conditioning of Diffusion Models

- Classifier guidance doesn't allow for arbitrary conditioning mechanisms.

What if we want to generate: $x_{new} \sim p(x_i|y)$ (where 'y' can be natural language text, image layout, etc.)



[SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis (Podell et al., 2023)]

Classifier-free Guidance

What if we want to generate: $x_{new} \sim p(x_i|y)$ (where 'y' can be natural language text, image layout, etc.)

Re-formulate:

$$p(y \mid x) = \frac{p(x \mid y) \cdot p(y)}{p(x)}$$

$$\log p(y \mid x) = \log p(x \mid y) + \log p(y) - \log p(x)$$

From classifier guidance:

$$\log p_\gamma(x|y) \propto \log p(x) + \gamma \boxed{\log p(y|x)}$$

$$\log p_\gamma(x|y) \propto \log p(x) + \gamma(\log p(x|y) + \cancel{\log p}(y) - \log p(x))$$

$$\boxed{\log p_\gamma(x|y) \propto (1-\gamma) \log p(x) + \gamma \log p(x|y)}$$

Train a conditional and an unconditional model in tandem.

Classifier-Free Diffusion Guidance (Ho et al., 2022)

Classifier-free Guidance

$$\log p_{\gamma}(x|y) \propto (1-\gamma)\log p(x) + \gamma \log p(x|y)$$

γ = 1.0 **γ = 3.0**

Prompt: A stained glass window of a panda eating bamboo.

Samples from OpenAI GLIDE text-to-image model (*Nichol et al., 2021*) with classifier-free guidance.

[Sander Dielemann's blog on Diffusion guidance]

Diffusion Models For Text

Why Diffusion Models for Text?

Autoregressive model: suffers from sampling drifts, major cause of hallucination

Humans do not necessarily process text sequentially.
Eg: *The food service of the restaurant was brilliant. The plate was cold and so was the food.*

Sequential generation does not take advantage of GPU parallelism (which can boost generation in long-contexts)

What are Diffusion Models doing Differently?

Non-autoregressive Generation: Generate all tokens in parallel

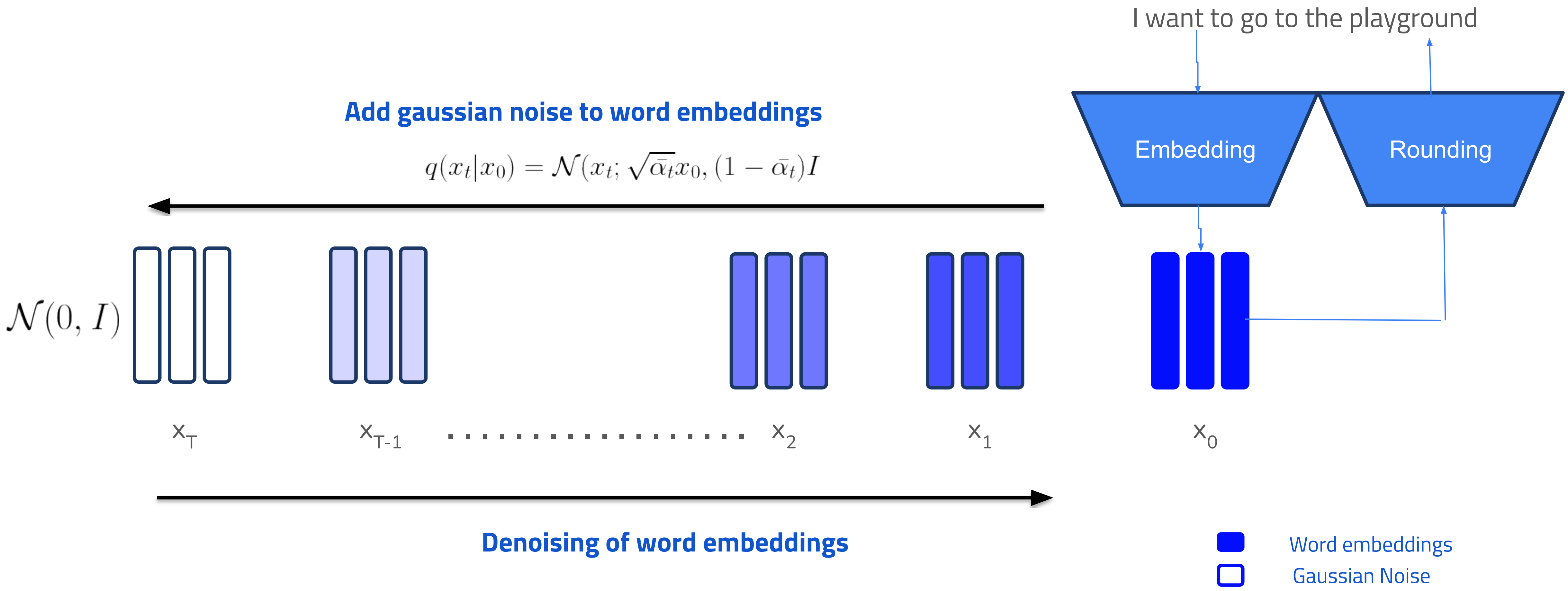
Iterative Denoising: allows reconsideration of already generated tokens

Exhibit better control

Continuous Diffusion Models For Text

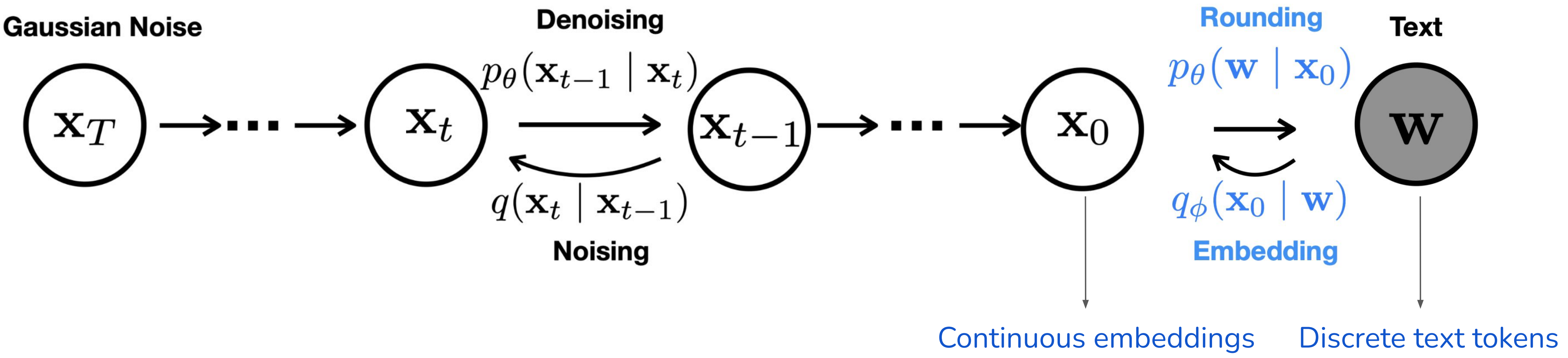
Continuous Diffusion for Text

- Gaussian diffusion processes defined on word embeddings.



[Diffusion-LM Improves Controllable Text Generation (Li et al., NeurIPS 2022)]

Continuous Diffusion for Text



- **Embedding step:** Maps discrete tokens to an embedding space.

$$q_\phi(x_0|w) = \mathcal{N}(\mathcal{E}(w), \sigma_0 I)$$

- **Rounding step:** Finds out the most likely token given an embedding (reverse of what embedding step does).

$$p_\theta(w|x_0) = \prod_{i=1}^{n} p_\theta(w_i|x_i)$$

(each component represents a position in the sentence modeled with a softmax function)

The rest of the forward process is defined in the same way as shown earlier.

[Diffusion-LM Improves Controllable Text Generation (Li et al., NeurIPS 2022)]

Training Continuous Diffusion for Text

$$\text{Loss} = \boxed{L_{VLB}} + \mathop{\mathbb{E}}_{q_\phi(x_0|w)} \boxed{\log \frac{q_\phi(x_0|w)}{p_\theta(w|x_0)}}$$

$$= \mathop{\mathbb{E}}_{x_t \sim q(x_t|x_0),\, x_0 \sim q_\phi(x_0|w)} \left[\boxed{\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2} + \boxed{\log q_\phi(x_0|w) - \log p_\theta(w|x_0)} \right]$$

$$= \mathop{\mathbb{E}}_{x_t \sim q(x_t|x_0),\, x_0 \sim q_\phi(x_0|w)} \left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 + \|\mathcal{E}(w) - \mu_\theta(x_1, 1)\|^2 - \log p_\theta(w|x_0) \right]$$

The first term can be reparameterized by letting the neural network directly predict $\mathbf{x_0}$ instead of predicting **noise**. Li et al., 2022 found that doing so reduced rounding errors.

$$= \mathop{\mathbb{E}}_{x_t \sim q(x_t|x_0),\, x_0 \sim q_\phi(x_0|w)} \left[\|\underbrace{h_\theta(x_t, t)}_{\text{predicted } x_0 \text{ at timestep } t} - x_0\|^2 + \|\mathcal{E}(w) - \mu_\theta(x_1, 1)\|^2 - \underbrace{\log p_\theta(w|x_0)}_{\text{Cross-entropy loss}} \right]$$

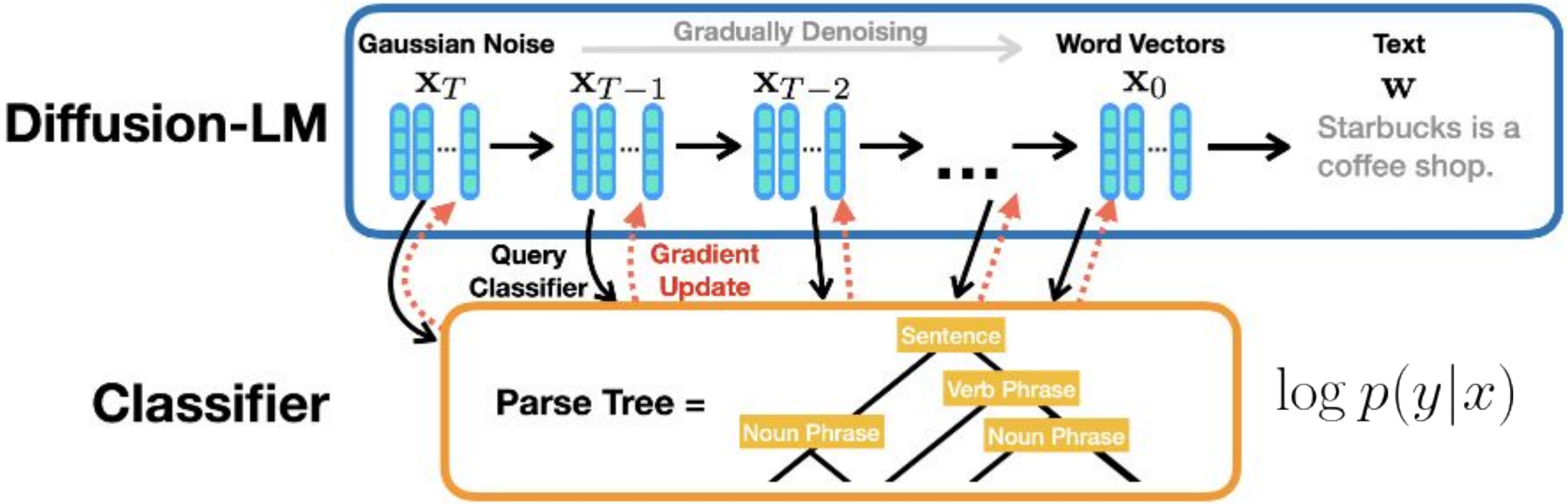
[Diffusion-LM Improves Controllable Text Generation (Li et al., NeurIPS 2022)]

Controllable Continuous Diffusion for Text

- We can develop controllable diffusion LMs with classifier guidance!

$$x_{new} \sim p(x_i|y)$$

('y' is a control signal)



$\log p(x)$

$\log p(y|x)$

$$\log p_\gamma(x|y) \propto \log p(x) + \gamma \log p(y|x)$$

[Diffusion-LM Improves Controllable Text Generation (Li et al., NeurIPS 2022)]

Controllable Continuous Diffusion for Text

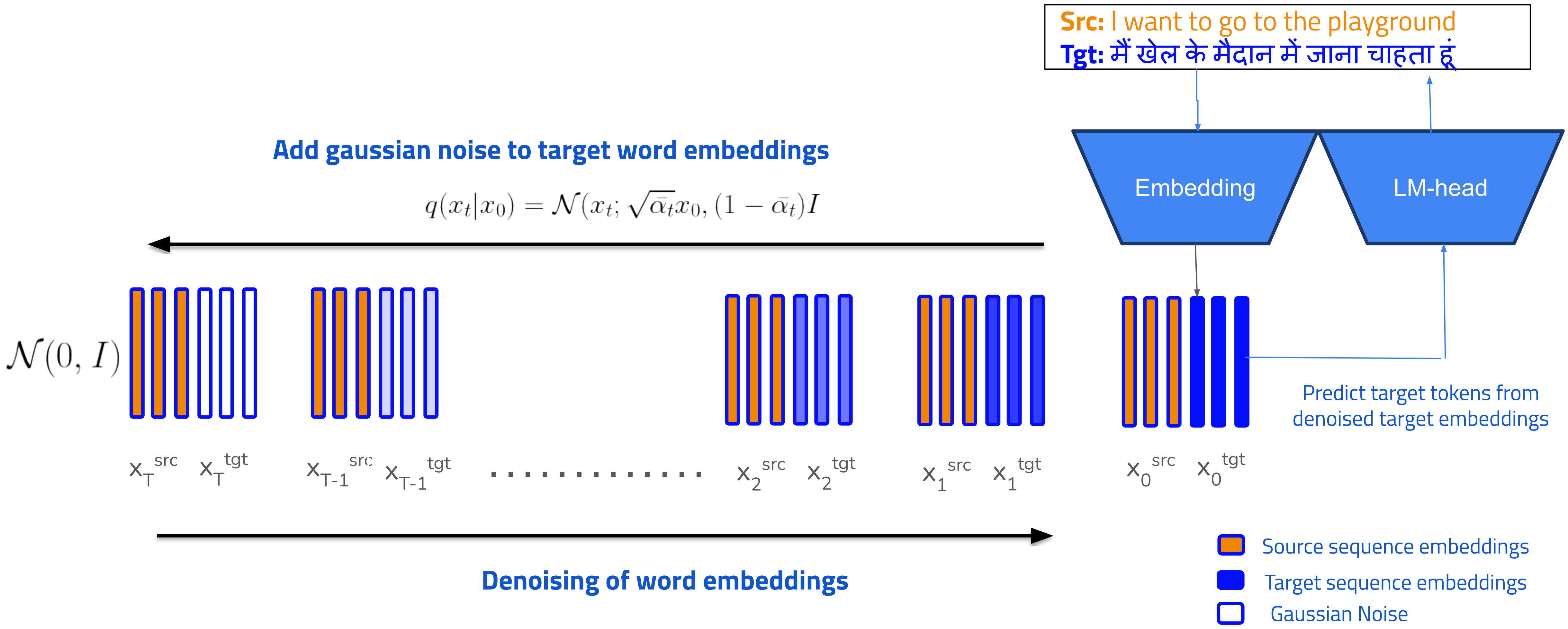
- We can develop controllable diffusion LMs with classifier guidance!

target POS	PROPN AUX DET ADJ NOUN NOUN VERB ADP DET NOUN ADP DET NOUN PUNCT
FUDGE	Aromi is a non family - friendly fast food coffee shop in the riverside area with a low Customer Rating .
Diffusion-LM	Fitzbillies is a cheap coffee shop located on the outskirts of the city .
FT	Aromi is a fast food pub located at the centre of the city.
target POS	PROPN AUX DET NOUN VERB NOUN ADJ NOUN PUNCT PRON NOUN NOUN AUX ADJ
FUDGE	Cocum is a family - friendly coffee shop , that has a low price range and a low customer rating .
Diffusion-LM	Zizzi is a pub providing restaurant Chinese food . Its customer rating is low
FT	Zizzi is a pub providing kids friendly services. Its customer rating is average

[Diffusion-LM Improves Controllable Text Generation (Li et al., NeurIPS 2022)]

DiffuSeq (Gong et al., 2022)

- Gaussian diffusion processes on word embeddings corresponding to the target sequence.



DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models (Gong et al., ICLR 2023)

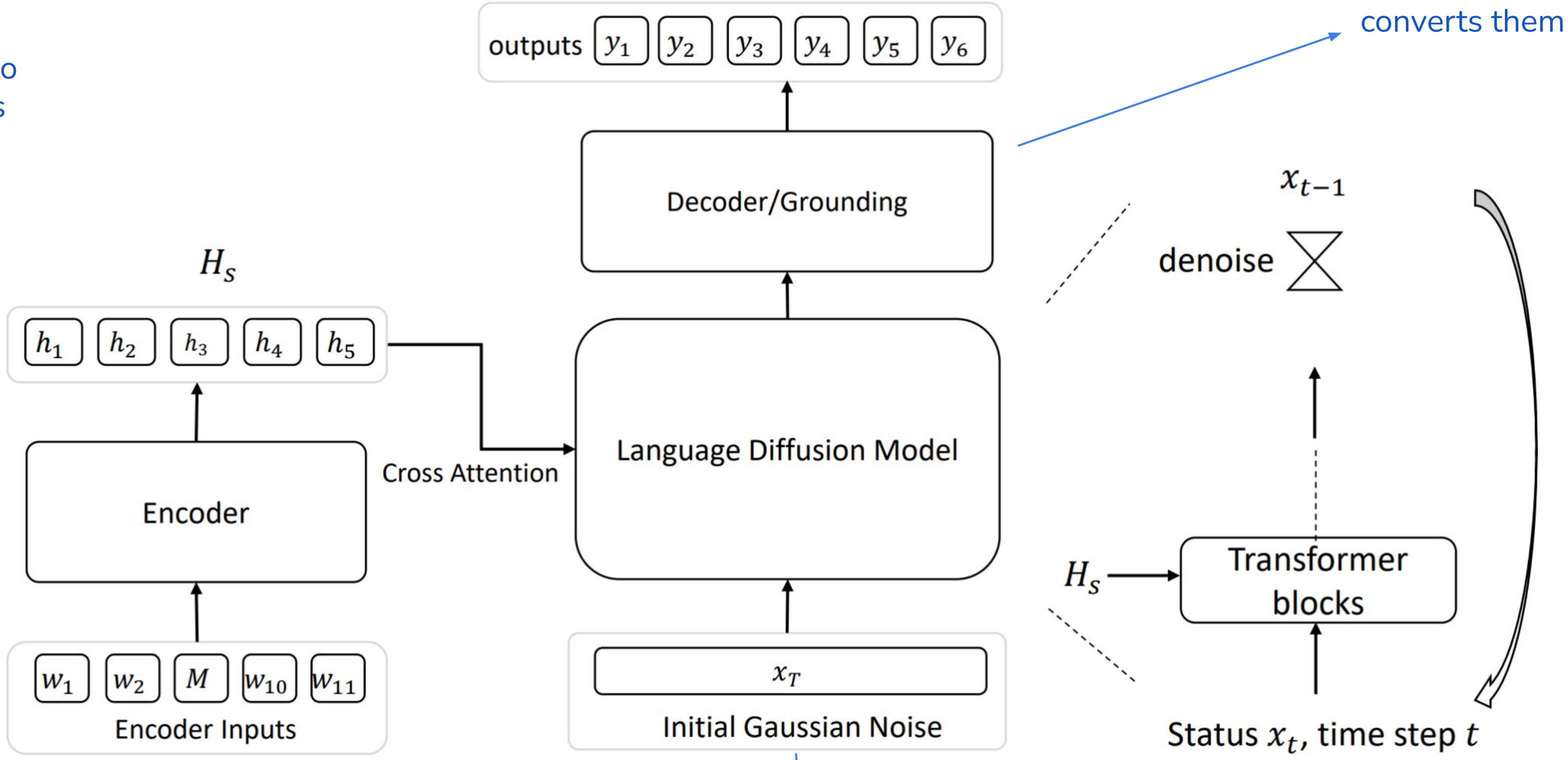
GENIE(Lin et al., 2023)

- Seq-to-seq generation with encoder-decoder style architecture.

Tgt: मैं खेल के मैदान में जाना चाहता हूं

Encodes input tokens to hidden representations

The decoder takes denoised output embeddings (i.e. x_0) as input and converts them to output tokens.



Src: I want to go to the playground

Denoiser or the language diffusion model is responsible for denoising the output representations.

[Text Generation with Diffusion Language Models: A Pre-training Approach with Continuous Paragraph Denoise (Lin et al., ICML 2023)]

Discrete Diffusion Language Models

Discrete Diffusion

- Can we directly denoise in the token space (which is discrete)?
- No need to define extra embedding and rounding steps which are harder to train jointly with the diffusion model.

Continuous Diffusion	**Discrete diffusion**
Add Gaussian noise (continuous distribution)	Add discrete noise (what is discrete noise)?
Diffusion process converges to a gaussian	Diffusion process converges to a categorical distribution (which categorical distribution)?

Masked Diffusion Language Models

- Defines the diffusion processes as iterative masking/unmasking of tokens.

$q(x_t|x_{t-1})$

Noising Process

What do you do in the morning? I go for a morning walk with my dog every day.

What do you do in the morning? I go [M] a morning walk with my dog every day.

What do you do in the morning? I go [M] a morning walk [M] my dog every day.

What do you do in the morning? I go [M] [M] morning walk [M] my [M] every day.

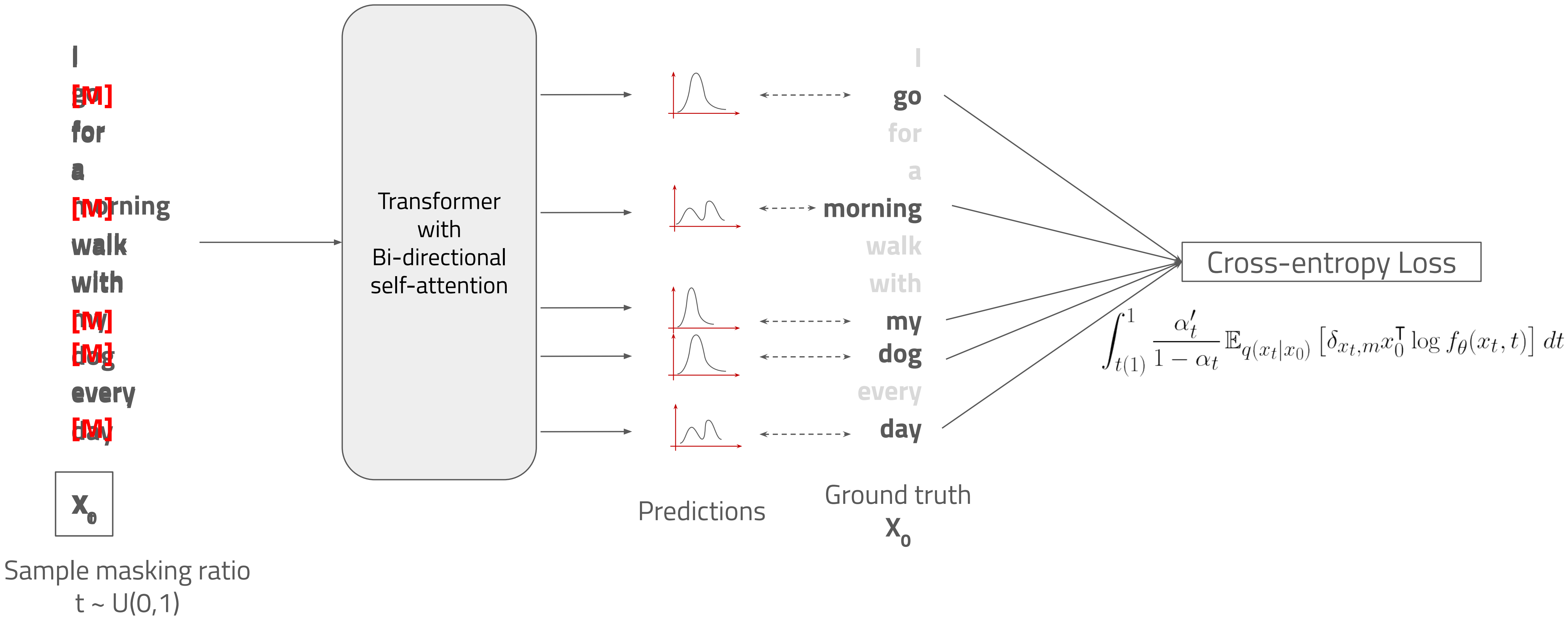
What do you do in the morning? [M] go [M] [M] [M] walk [M] my [M] [M] [M].

What do you do in the morning? [M] [M] [M] [M] [M] [M] [M] [M] [M] [M] [M].

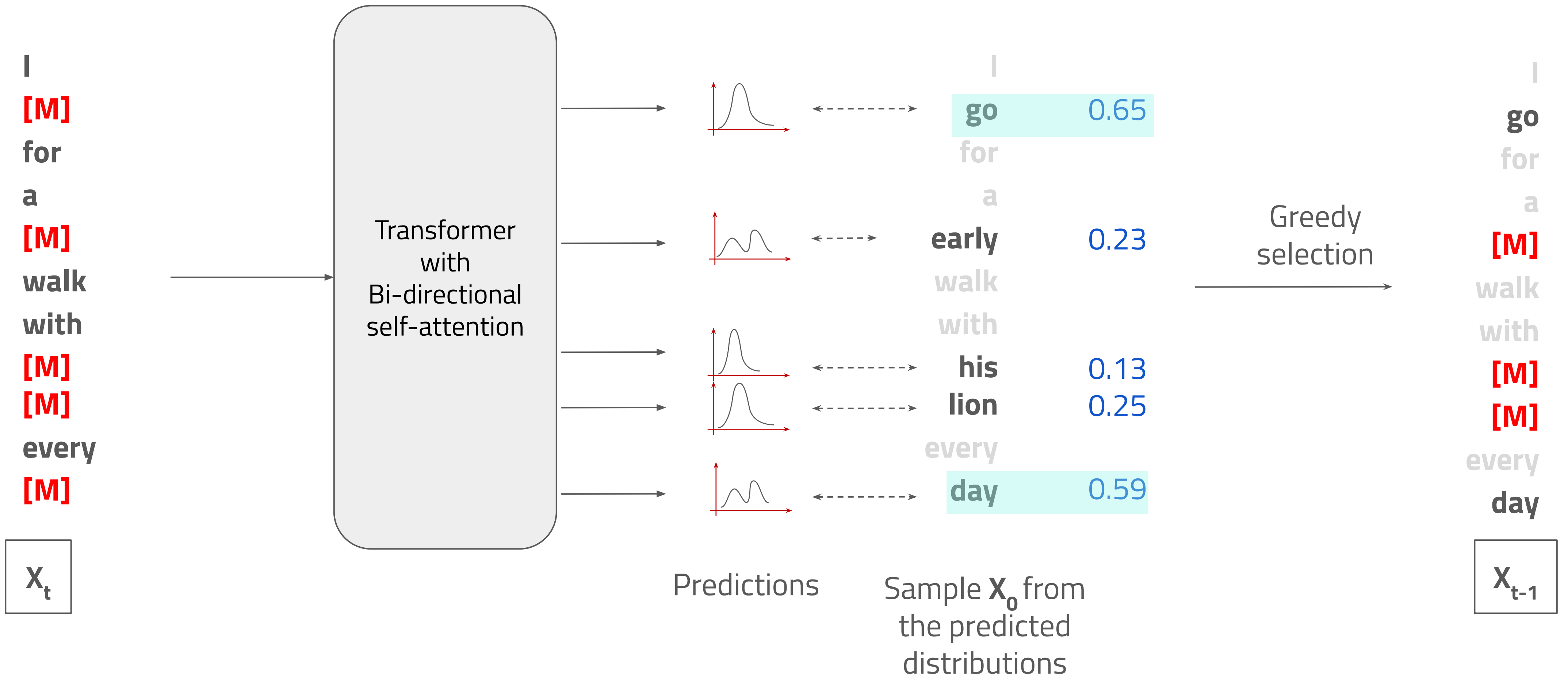
$p_\theta(x_{t-1}|x_t)$

Generative Process

Training Masked Diffusion Language Models



Greedy Sampling Step in Masked Diffusion Language Models



Uniform Diffusion Language Models

Vocabulary

A, ability, able, ...

.
.
.
.
.
.
.
.

.... z, zeta

Random replacement probability

$q(x_t|x_{t-1})$

p = 0	I go for a morning walk with my dog every day.
p = 0.1	I go **not** a morning walk with my dog every day.
p = 0.2	I go **not** a morning walk **safe** my dog every day.
p = 0.3	I go **not ant** morning walk **safe** my **car** every day.
p = 1.0	**Have zeta not ant cross hell safe human car a donut.**

$p_\theta(x_{t-1}|x_t)$

Every position in this sentence is simply a uniform randomly sampled word from the vocabulary

UDLM: Formal Definition

$$\text{Let } x = < x^{(1)}, x^{(2)}, \ldots\ldots, x^{(n)} >$$ (sequence of tokens)

Let x_0 be a one-hot vector of length equal to vocabulary size V

$$q(x_t^i | x_0^i) = \alpha_t x_0^i + (1-\alpha_t)\pi, \text{where } \pi = \frac{\mathbb{I}}{|V|}$$

Each value in vector **π** is **1/|V|** **Forward process**

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

$$f_\theta(x_t, t) = \mathbf{NN}_\theta(x_t, t)$$

True reverse

Estimate of x_0

$$p_\theta(x_{t-1} | x_t) = q(x_{t-1} | x_t, f_\theta(x_t, t))$$

Loss Function: $\mathcal{L}^\infty = \int_{t=0}^{t=1} \mathbb{E}_q \left[\frac{\alpha_t'}{N\alpha_t} \left[\frac{N}{\bar{\mathbf{x}}_i} - \frac{N}{(\bar{\mathbf{x}}_\theta)_i} - \sum_{j \text{ s.t. } (\mathbf{z}_t)_j = 0} \left(\frac{\bar{\mathbf{x}}_j}{\bar{\mathbf{x}}_i} \right) \log \left[\left(\frac{(\bar{\mathbf{x}}_\theta)_i \cdot \bar{\mathbf{x}}_j}{(\bar{\mathbf{x}}_\theta)_j \cdot \bar{\mathbf{x}}_i} \right) \right] \right] \right] \mathrm{d}t$ 🥲

Simple Guidance Mechanisms for Discrete Diffusion Models (Schiff et al., 2024)

Diffusion vs. Autoregression

Diffusion vs. Autoregression

Autoregressive LMs	Diffusion LMs
Sequential generation	Non-autoregressive generation with iterative denoising
Harder to control, exhibit sampling drifts.	Easier to control
Lot of innovation (Eg: KV-caching, group-query attention, etc.)	Nascent stage of research, not completely practical yet

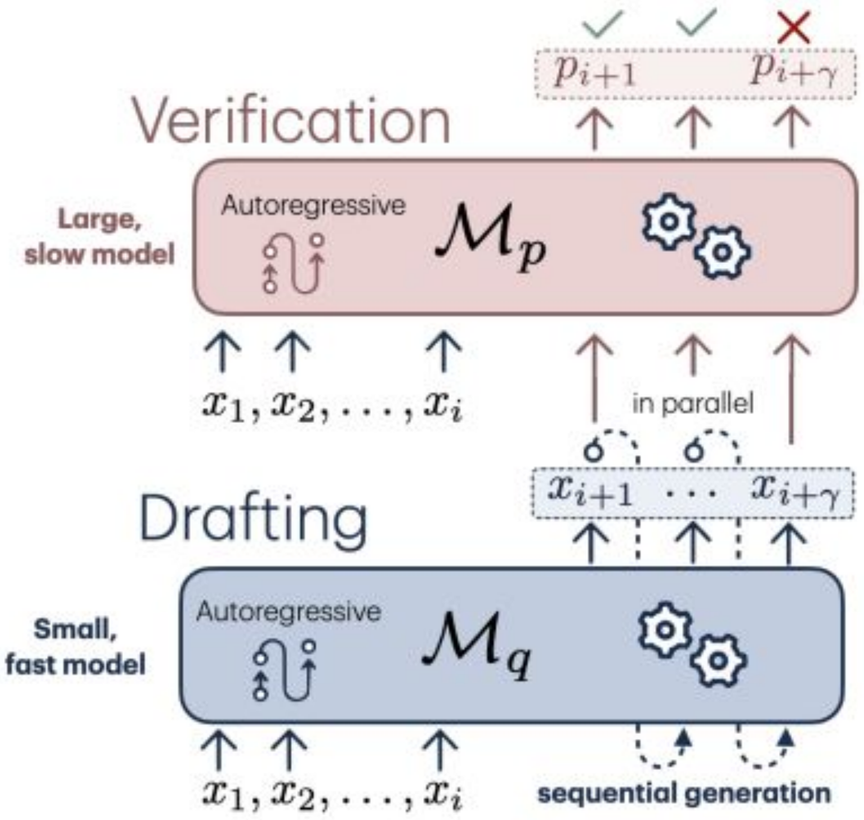
Can we combine them?

Can we Combine Autoregressive LLMs and Diffusion LMs?

- Why combine? To try to make use of the qualities of both.
- **Challenge:** Completely different training paradigms!
- **Ways to combine:**
  - Diffusion model as a helper for LLM
  - Adapting an LLM to a diffusion model

Speculative Decoding

- Employs a small LM to help parallelize token generation from a larger model.



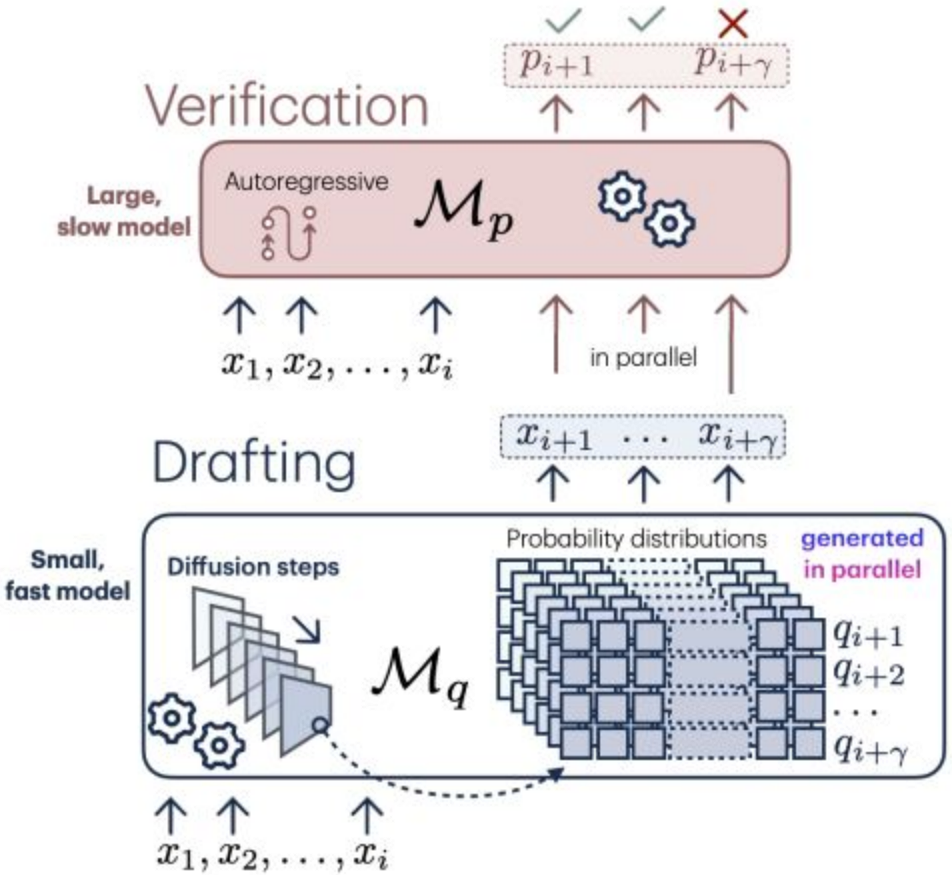
Verify candidate sequences in parallel with a larger model

Generate candidate sequences sequentially from a smaller but faster model

[Speculative Diffusion Decoding: Accelerating Language Generation through Diffusion (Christopher et al., 2024)]

Speculative Diffusion Decoding

- Use a small diffusion LM to generate potential candidates.



Quality vs speed trade-off can be done by tuning the diffusion steps

Generate candidate sequences non-autoregressively with a small diffusion model

[Speculative Diffusion Decoding: Accelerating Language Generation through Diffusion (Christopher et al., 2024)]

Adapting LLMs to Diffusion LMs

- Given that we have a lot of pre-trained autoregressive LLMs, can we take advantage of their pre-training and try adapting them to diffusion LMs?
- Naive fine-tuning won't work due to difference in training paradigms.
- How to bring these training paradigms closer?
- *Gong et al., 2024* proposes masked diffusion LMs as a solution!

Adapting LLMs to Masked Diffusion LMs

- Previously, we observed that masked diffusion LM loss can be simplified to cross-entropy.

- Cross-entropy loss is also used to train pre-trained LMs.

(Masked Diffusion Loss)

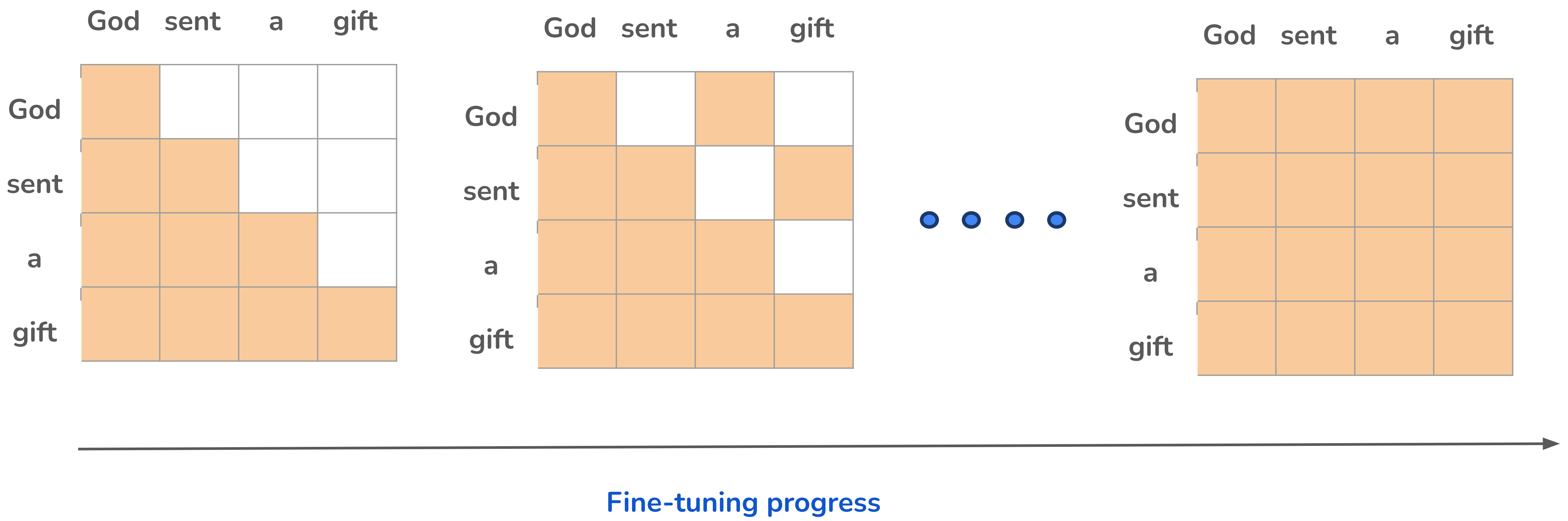
$$\mathcal{L}_T = \int_0^1 \frac{\alpha'_t}{1-\alpha_t} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [\delta_{\boldsymbol{x}_t, \boldsymbol{m}} \boldsymbol{x}_0^\top \log f_\theta(\boldsymbol{x}_t)] \, dt.$$

(LLM Loss)

$$\mathcal{L}_{AR}^{1:N} = -\sum_{n=1}^{N} (\mathbf{x}_0^n)^\top \log f_\theta(\mathbf{x}_0^{1:n-1})_{n-1}$$

- **How do we adapt?**
  - **Issue no .1:** LLM uses a causal attention mask while diffusion uses bi-drectional attention.
  - **Issue no. 2:** LLMs predicts for the next position while diffusion predicts on the same one.

Causal Attention to Bi-directional Attention



Scaling Diffusion Language Models via Adaptation from Autoregressive Models (Gong et al., 2024)

Training Algorithm

Algorithm 1 Adaptation Training

Input: network f_θ initialized by existing models, training corpus $p_{data}(\boldsymbol{x}_0^{1:N})$, mask token $\boldsymbol{m}$.
Output: model parameters θ.
repeat

Forward diffusion step for masked diffusion

  Draw $\boldsymbol{x}_0^{1:N} \sim p_{data}$ and set *labels* $\leftarrow \boldsymbol{x}_0^{1:N}$
  Sample $t \in \mathit{Uniform}(0, 1)$
  Sample $\boldsymbol{x}_t^{1:N} \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0)$

  Anneal the attention mask $attn\_mask$
  Forward *logits* $\leftarrow f_\theta(\boldsymbol{x}_t^{1:N})$ with $attn\_mask$
  Right shift *logits* by one position

Masked diffusion loss:

  $\mathcal{L}_t = \frac{1}{t}\delta_{x_t,m}\mathrm{CE}(\mathit{logits}, \mathit{labels})$ ▷ Eq.7
  Backprop with $\mathcal{L}_t$ and update θ
until end training

[Scaling Diffusion Language Models via Adaptation from Autoregressive Models (Gong et al., 2024)]

Supplementary Material

- DDPM blog: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- Score SDE blog: https://fanpu.io/blog/2023/score-based-diffusion-models/
- Flow matching and Diffusion Models lectures: https://youtube.com/playlist?list=PL57nT7tSGAAUDnli1LhTOoCxlEPGS19vH&si=L8T3fjOFbhooLj9T

Thank You