Diffusion-LM Improves Controllable Text Generation

Xiang Lisa Li
Stanford University
xlisali@stanford.edu

John Thickstun
Stanford University
jthickst@stanford.edu

Ishaan Gulrajani
Stanford Univeristy
igul@stanford.edu

Percy Liang
Stanford Univeristy
pliang@cs.stanford.edu

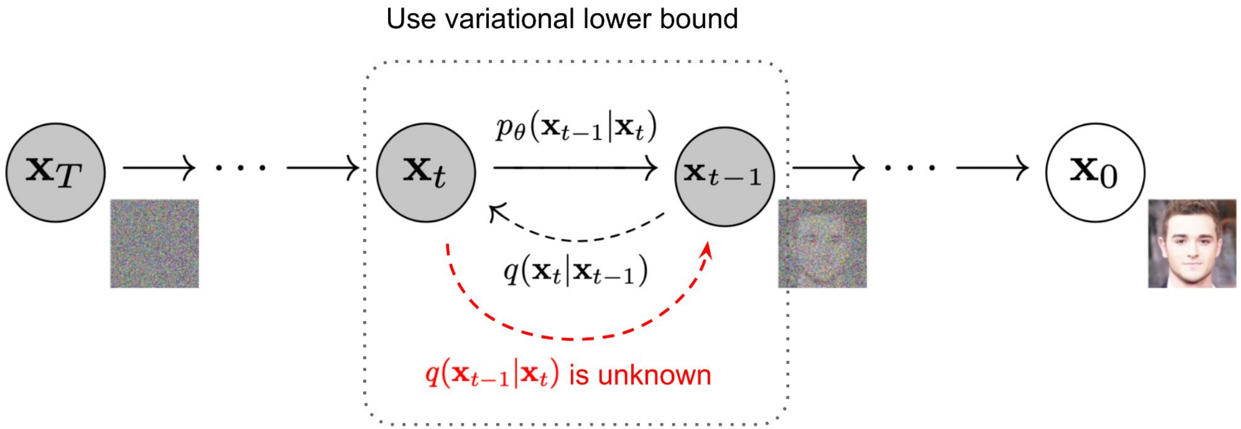
Tatsunori B. Hashimoto
Stanford Univeristy
thashim@stanford.edu

Introduction

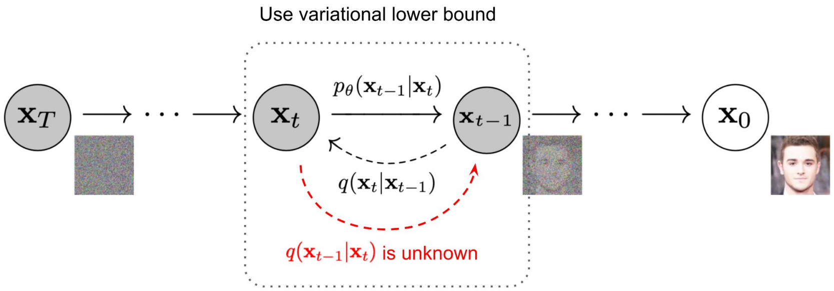
- Controllable Text Generation
  - Exploit large autoregressive language models
  - Limited to simple, attribute-level controls (e.g., sentiment or topic)
- This paper considers six control targets ranging from fine-grained attributes (e.g., semantic content) to complex structures (e.g., parse trees).
- They propose Diffusion-LM, a new language model based on continuous diffusions.
  - Diffusion-LM starts with a sequence of Gaussian noise vectors and incrementally denoises them into vectors corresponding to words.

General Diffusion Models

A diffusion model [12, 27] is a latent variable model that models the data $\mathbf{x}_0 \in \mathbb{R}^d$ as a Markov chain $\mathbf{x}_T \ldots \mathbf{x}_0$ with each variable in $\mathbb{R}^d$, and $\mathbf{x}_T$ is a Gaussian. The diffusion model incrementally denoises the sequence of latent variables $\mathbf{x}_{T:1}$ to approximate samples from the target data distribution (Figure 2). The initial state $p_\theta(\mathbf{x}_T) \approx \mathcal{N}(0, \mathbf{I})$, and each denoising transition $\mathbf{x}_t \rightarrow \mathbf{x}_{t-1}$ is parametrized by the model $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$. For example, μ_θ and Σ_θ may be computed by a U-Net or a Tranformer.



Forward diffusion process



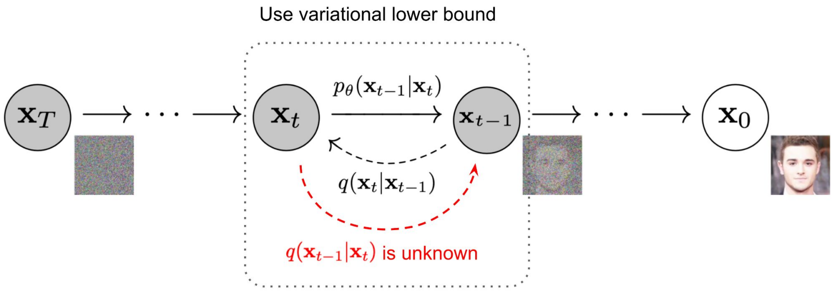
Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, let us define a *forward diffusion process* in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \dots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

The data sample $\mathbf{x}_0$ gradually loses its distinguishable features as the step t becomes larger. Eventually when $T \to \infty$, $\mathbf{x}_T$ is equivalent to an isotropic Gaussian distribution.

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Reverse diffusion process



If we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because it needs to use the entire dataset and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the *reverse diffusion process*.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

μ_θ and Σ_θ may be computed by a U-Net or a Tranformer.

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Training of Diffusion Model

The diffusion model is trained to maximize the marginal likelihood of the data $\mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}}[\log p_\theta(\mathbf{x}_0)]$, and the canonical objective is the variational lower bound of $\log p_\theta(\mathbf{x}_0)$ [39],

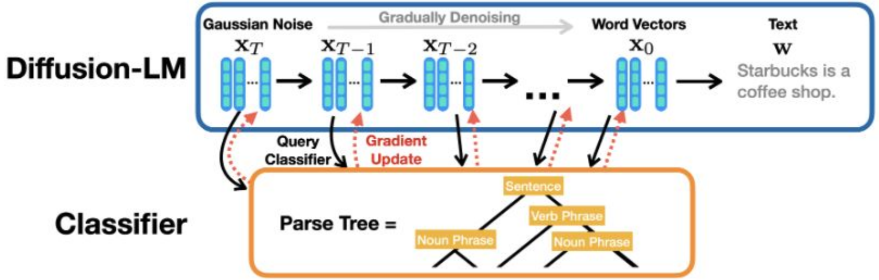
$$\mathcal{L}_{\text{vlb}}(\mathbf{x}_0) = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right]. \quad (1)$$

$$\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} ||\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)||^2,$$

where $\hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)$ is the mean of the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$ which is a closed from Gaussian, and $\mu_\theta(\mathbf{x}_t, t)$ is the predicted mean of $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ computed by a neural network.

Problem Statement

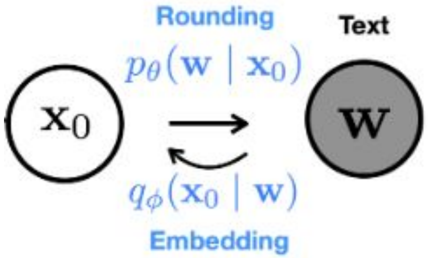
Text generation is the task of sampling $\mathbf{w}$ from a trained language model $p_{\text{lm}}(\mathbf{w})$, where $\mathbf{w} = [w_1 \cdots w_n]$ is a sequence of discrete words and $p_{\text{lm}}(\mathbf{w})$ is a probability distribution over sequences of words. Controllable text generation is the task of sampling $\mathbf{w}$ from a conditional distribution $p(\mathbf{w} \mid \mathbf{c})$, where $\mathbf{c}$ denotes a *control* variable. For syntactic control, $\mathbf{c}$ can be a target syntax tree (Figure 1), while for sentiment control, $\mathbf{c}$ could be a desired sentiment label. The goal of controllable generation is to generate $\mathbf{w}$ that satisfies the control target $\mathbf{c}$.



Diffusion-LM: Continuous Diffusion Language Modeling

1. An **embedding function** maps discrete text into a continuous space

$$\mathrm{EMB}(\mathbf{w}) = [\mathrm{EMB}(w_1), \ldots, \mathrm{EMB}(w_n)] \in \mathbb{R}^{nd}$$



The training objectives introduced in §3 now becomes

$$\begin{aligned}
\mathcal{L}^{\text{e2e}}_{\text{vlb}}(\mathbf{w}) &= \underset{q_\phi(\mathbf{x}_0|\mathbf{w})}{\mathbb{E}} \left[\mathcal{L}_{\text{vlb}}(\mathbf{x}_0) + \log q_\phi(\mathbf{x}_0|\mathbf{w}) - \log p_\theta(\mathbf{w}|\mathbf{x}_0)\right], \\
\mathcal{L}^{\text{e2e}}_{\text{simple}}(\mathbf{w}) &= \underset{q_\phi(\mathbf{x}_{0:T}|\mathbf{w})}{\mathbb{E}} \left[\mathcal{L}_{\text{simple}}(\mathbf{x}_0) + ||\mathrm{EMB}(\mathbf{w}) - \mu_\theta(\mathbf{x}_1, 1)||^2 - \log p_\theta(\mathbf{w}|\mathbf{x}_0)\right].
\end{aligned} \tag{2}$$

2. Reducing Rounding Errors

Empirically, the model fails to generate x0 that commits to a single word.

derive an analogue to $\mathcal{L}_{\text{simple}}$ which is parametrized via $\mathbf{x}_0$,

$$\mathcal{L}^{\text{e2e}}_{\mathbf{x}_0\text{-simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t} || f_\theta(\mathbf{x}_t, t) - \mathbf{x}_0||^2$$

where our model $f_\theta(\mathbf{x}_t, t)$ predicts $\mathbf{x}_0$ directly

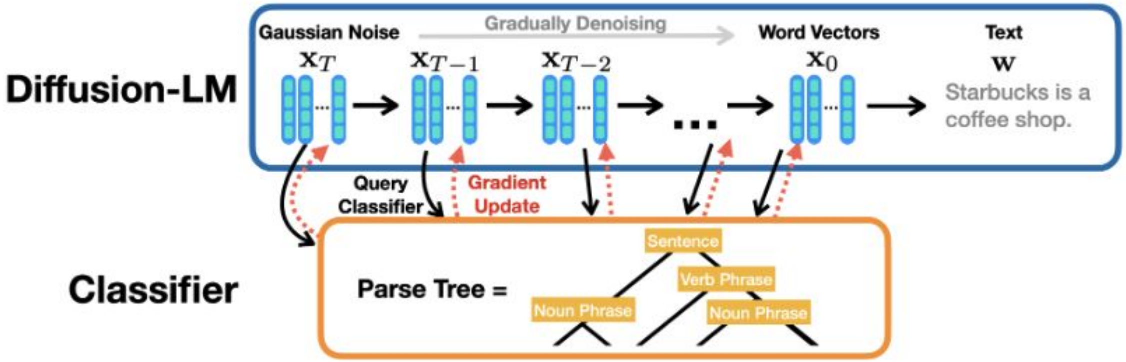
Decoding and Controllable Generation

At each diffusion step:

step: $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) \propto p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \cdot p(\mathbf{c} \mid \mathbf{x}_{t-1}, \mathbf{x}_t)$. We further simplify $p(\mathbf{c} \mid \mathbf{x}_{t-1}, \mathbf{x}_t) = p(\mathbf{c} \mid \mathbf{x}_{t-1})$ via conditional independence assumptions from prior work on controlling diffusions [40]. Consequently, for the t-th step, we run gradient update on $\mathbf{x}_{t-1}$:

$$\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) + \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{c} \mid \mathbf{x}_{t-1}),$$

where both $\log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ and $\log p(\mathbf{c} \mid \mathbf{x}_{t-1})$ are differentiable: the first term is parametrized by Diffusion-LM, and the second term is parametrized by a neural network classifier.



Generate Fluent Text

we run gradient updates on a control objective with fluency regularization:

$$\lambda \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) + \log p(\mathbf{c} \mid \mathbf{x}_{t-1})$$

We run 3 steps of the Adagrad update for each diffusion steps. To mitigate for the increased computation cost, we downsample the diffusion steps from 2000 to 200, which speeds up our controllable generation algorithm without hurting sample quality much

Minimum Bayes Risk Decoding

Select the sample that achieves the minimum expected risk under a loss function *L* (e.g., negative BLEU score).

$$\hat{\mathbf{w}} = \operatorname{argmin}_{\mathbf{w} \in S} \sum_{\mathbf{w}' \in S} \frac{1}{|S|} \mathcal{L}(\mathbf{w}, \mathbf{w}')$$

Experiments

Train Diffusion-LM on **two datasets**: E2E and ROCStories

The E2E dataset consists of 50K restaurant reviews labeled by 8 fields including food type, price, and customer rating.

The ROCStories dataset consists of 98K five-sentence stories, capturing a rich set of causal and temporal commonsense relations between daily events.

Diffusion-LM is based on **Transformer** and with a sequence length n = 64, diffusion steps T = 2000.

Decoding Diffusion-LM for 200 steps is still 7x slower than decoding autoregressive LMs.

Control tasks

- Semantic Content: field (e.g., rating) and value (e.g., 5 star)
- Parts-of-speech
- Syntax Tree
- Syntax Spans
- Length
- Infilling

Baselines

- Classifier-Guided Control Baselines
  - PPLM
  - FUDGE
  - FT
- Infilling Baselines
  - DELOREAN
  - COLD
  - AR-infilling

Classifier-Guided Controllable Text Generation Results

	Semantic Content		Parts-of-speech		Syntax Tree		Syntax Spans		Length	
	ctrl ↑	lm ↓	ctrl ↑	lm ↓	ctrl ↑	lm ↓	ctrl ↑	lm ↓	ctrl ↑	lm ↓
PPLM	9.9	5.32	-	-	-	-	-	-	-	-
FUDGE	69.9	2.83	27.0	7.96	17.9	**3.39**	54.2	4.03	46.9	3.11
Diffusion-LM	**81.2**	**2.55**	**90.0**	**5.16**	**86.0**	3.71	**93.8**	**2.53**	**99.9**	**2.16**
FT-sample	72.5	2.87	89.5	4.72	64.8	5.72	26.3	2.88	98.1	3.84
FT-search	89.9	1.78	93.0	3.31	76.4	3.24	54.4	2.19	100.0	1.83

Table 2: Diffusion-LM achieves high success rate (ctrl ↑) and good fluency (lm ↓) across all 5 control tasks, outperforming the PPLM and FUDGE baselines. Our method even outperforms the fine-tuning oracle (FT) on controlling syntactic parse trees and spans.

Composition of Controls

	Semantic Content + Syntax Tree			Semantic Content + Parts-of-speech		
	semantic ctrl ↑	syntax ctrl ↑	lm ↓	semantic ctrl ↑	POS ctrl ↑	lm ↓
FUDGE	61.7	15.4	3.52	64.5	24.1	3.52
Diffusion-LM	**69.8**	**74.8**	5.92	**63.7**	**69.1**	3.46
FT-PoE	61.7	29.2	**2.77**	29.4	10.5	**2.97**

Table 4: In this experiment, we compose semantic control and syntactic control: Diffusion-LM achieves higher success rate (ctrl ↑) at some cost of fluency (lm ↓). Our method outperforms both FUDGE and FT-PoE (product of experts of two fine-tuned models) on control success rate, especially for the structured syntactic controls (i.e. syntactic parse tree and POS).

Infilling Results

	Automatic Eval				Human Eval
	BLEU-4 ↑	ROUGE-L ↑	CIDEr ↑	BERTScore ↑	
Left-only	0.9	16.3	3.5	38.5	n/a
DELOREAN	1.6	19.1	7.9	41.7	n/a
COLD	1.8	19.5	10.7	42.7	n/a
Diffusion	**7.1**	**28.3**	**30.7**	**89.0**	$\mathbf{0.37}^{+0.03}_{-0.02}$
AR	6.7	27.0	26.9	**89.0**	$\mathbf{0.39}^{+0.02}_{-0.03}$

Table 5: For sentence infilling, Diffusion-LM significantly outperforms prior work COLD [31] and Delorean [30] (numbers taken from paper), and matches the performance of an autoregressive LM (AR) trained from scratch to do infilling.

Ablation Studies

Learned v.s. Random Embeddings. Learned embeddings outperform random embeddings

Objective Parametrization. Parametrizing by X0 consistently attains good performance across dimensions.

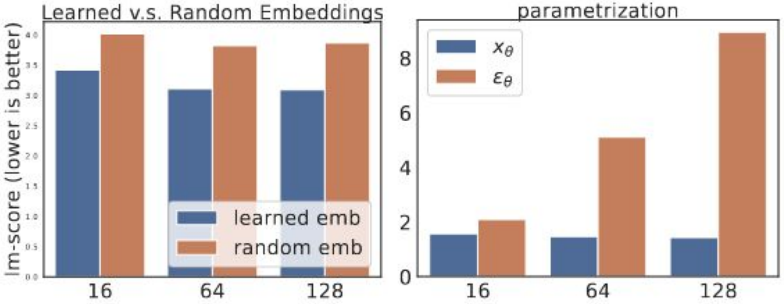


Figure 4: We measure the impact of our proposed design choices through lm-score. We find both learned embeddings and reparametrization substantially improves sample quality.