Diffusion-LM Improves Controllable Text Generation

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, Tatsunori B. Hashimoto
Stanford University
Motivating Diffusion-LM

(Dhariwal and Nichol, 2021)

Diffusion models are now dominant in vision. Are they also good for language?
Motivating Diffusion-LM: Classifier Guidance

- Diffusion models can be easily and convincingly steered using a probabilistic scoring function (e.g., a classifier).

- Analogous to “plug-and-play” language modeling (Dathathri et al., 2020).

- This post-hoc conditioning seems compelling:
  - train one general-purpose (expensive) generative model.
  - steer it for your specialized task at inference time.

(Dhariwal and Nichol, 2021)
Motivating Diffusion-LM

Classifier Guidance vs. Prompting: competing or complementary paradigms?

(Ramesh et al., 2022)
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Control

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Control

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Control

• Experiments

• Discussion
Denoising Diffusion Probabilistic Models

- Learn to generate data by progressive denoising.
Denoising Diffusion Probabilistic Models

- **Forward Process.** Given *noise schedule* \( \beta_1, \ldots, \beta_T \) (hyper-parameters):

  \[
  q(x_t | x_{t-1}) = \mathcal{N} \left( x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I \right).
  \]

- **Reverse Process.** Learn to denoise with parameters \( \theta \):

  \[
  p_\theta(x_{t-1} | x_t) = \mathcal{N} \left( x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t) \right).
  \]
Optimizing a Diffusion Model

- Optimize like a VAE (the usual variational lower bound):
  \[
  - \log p_\theta(x_0) = - \log \int_{x_1:T} p_\theta(x_{0:T}) \, dx_{1:T} = - \log \mathbb{E}_{x_1:T \sim q} \left[ \frac{p_\theta(x_{0:T})}{q(x_{1:T} | x_0)} \right] \\
  \leq \mathbb{E}_{x_1:T \sim q} \left[ - \log \frac{p_\theta(x_{0:T})}{q(x_{1:T} | x_0)} \right] = \mathbb{E}_{x_1:T \sim q} \left[ - \log q(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} | x_t)}{q(x_t | x_{t-1})} \right].
  \]
- Many small steps $\beta_t$ make the Gaussian approximation $p_\theta(x_{t-1} | x_t)$ valid.
In What Sense is this Denoising?

- Rewrite the variational objective (algebra; see Ho et al., 2020; Appendix A):

\[
\mathbb{E}_{x_{1:T} \sim q} \left[ D(q(x_T|x_0) \| p(x_T)) + \sum_{t=1}^{T} D(q(x_{t-1}|x_t,x_0) \| p_\theta(x_{t-1}|x_t)) - \log p_\theta(x_0|x_1) \right].
\]

- Want to make \( p_\theta(x_{t-1}|x_t) \) look like the posterior distribution \( q(x_{t-1}|x_t,x_0) \).
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Talk Outline

• **Constructing Diffusion-LM**
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ **Learning Word Embeddings (End-to-End Training)**
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Learning Word Embeddings

- Standard DDPM assumes inputs $x_0$ are continuous.
- Language model inputs $w$ are discrete tokens.
- Use the re-parameterization trick to learn word embeddings $q_\phi(x_t | w)$:

$$- \log p_\theta(x_0) \leq \mathbb{E}_{x_1:T \sim q, x_0 \sim q_\phi} \left[ - \log q(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} | x_t)}{q(x_t | x_{t-1})} - \log p_\theta(w | x_0) \right].$$
Are These Learned Embeddings Meaningful?

• Learning the embedding seems important.

• Random embeddings performed poorly.

• Pre-trained embeddings from an AR model also performed poorly.

A t-SNE plot of learned embeddings, colored according to POS.
Talk Outline

• **Constructing Diffusion-LM**
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Predicting the Noiseless Embeddings

- We want to minimize terms $D(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$, where
  
  $$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)),$$

- Closed form for the posteriors: $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$, where
  
  $$\tilde{\mu}_t(x_t, x_0) = r_t x_t + s_t x_0.$$

- And $r_t, s_t, \tilde{\beta}_t$ are constants derived from the noise schedule $\beta_1, \ldots, \beta_T$. 

Diffusion-LM

Reparameterization
Predicting the Noiseless Embeddings

• Directly parameterize $\mu_\theta(x_t, t)$ to approximate $\tilde{\mu}_t(x_t, x_0) = r_t x_t + s_t x_0$?

• But we already know $x_t$.

• Ho et al., 2020: write $x_0 = x_t - \epsilon_t$, predict $\epsilon_t \approx \epsilon_\theta(x_t, t)$ and reparameterize

$$\mu_\theta(x_t, t) = (r_t + s_t)x_t - s_t\epsilon_\theta(x_t, t).$$

• Li et al., 2022: predict $x_0 \approx f_\theta(x_t, t)$ and reparameterize

$$\mu_\theta(x_t, t) = r_t x_t + s_t f_\theta(x_t, t).$$
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• **Sampling from Diffusion-LM**
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Sampling from Diffusion-LM

- Sample from a Gaussian $x_T \sim \mathcal{N}(0, I)$ and then just follow the chain.

- Sampling Diffusion-LM requires $T$ (diffusion steps) model calls.

- In contrast, sampling AR models requires $L$ (sequence length) model calls.

- In our experiments, $T \gg L$ so sampling is slow.
Sampling Heuristics

- Analogous AR sampling heuristics can be applied to Diffusion-LM.

- Temperature sampling: reduce the noise at each sampling step.

- Nucleus sampling: truncate the tails of the Gaussian noise.

- Minimum Bayes Risk (MBR) decoding.
The Clamping Trick

- At each step we predict $x_0 \approx f_\theta(x_t, t)$.

- *The clamping trick*: instead predict the nearest embedding in the dictionary:

  $x_0 \approx \text{Clamp}(f_\theta(x_t, t))$.

- This seems to nudge Diffusion-LM to commit to tokens earlier.
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Classifier-Guided Sampling

• Given: labeled data pairs \((x, c)\) describing the desired attribute \(c\).

• Goal: sample from the posterior distribution

\[
p_{\theta}(x_{0:T}|c) = \prod_{t=1}^{T} p_{\theta}(x_{t-1}|x_{t}, c).
\]

• Each term can be rewritten as

\[
p_{\theta}(x_{t-1}|x_{t}, c) \propto p_{\theta}(c|x_{t-1}, x_{t}) p_{\theta}(x_{t-1}|x_{t}).
\]

• The term \(p_{\theta}(c|x_{t-1}, x_{t})\) is a classifier. And \(p_{\theta}(x_{t-1}|x_{t})\) is Diffusion-LM.

• In practice, the classifier doesn’t need to see both \(x_{t}\) and \(x_{t-1}\).

• Train the classifier \(p_{\theta}(c|x_{t-1})\) on noisy data.
Langevin Dynamics

• Goal: sample from the posterior $p_\theta(x_{t-1}|x_t, c)$.

• Langevin Dynamics: define a Markov chain

\[
x^{(i+1)}_{t-1} = x^{(i)}_{t-1} - \eta \nabla_{x_{t-1}} \log p_\theta(x_{t-1}|x_t, c) + \sqrt{2\eta} \varepsilon_i, \text{ where } \varepsilon_i \sim \mathcal{N}(0, I).
\]

• For small $\eta$, as $i \to \infty$, $D(x^{(i+1)}_{t-1} \parallel p_\theta(x_{t-1}|x_t, c)) \to 0$. The gradient is:

\[
\nabla_{x_{t-1}} \log p_\theta(x_{t-1}|x_t, c) = \nabla_{x_{t-1}} \log p_\theta(c|x_{t-1}) + \nabla_{x_{t-1}} \log p_\theta(x_{t-1}|x_t).
\]

• In practice, init $x^{(0)}_{t-1} \sim p_\theta(x_{t-1}, x^{(1)}_t)$ (warmstart) and take just one step.
Iterative Gradient-Based Control

- Conceptually similar to PPLM (Dathathri et al., 2020).
- But control is applied coarse-to-fine, instead of left-to-right.
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Datasets

- Two datasets: **E2E** and **ROCStories**.

- **E2E.** 50k restaurant reviews. Sample text: “Browns Cambridge is good for Japanese food and also children friendly near The Sorrento.”

- **ROCStories.** 98k short stories. Sample text: "Jennifer has a big exam tomorrow. She got so stressed, she pulled an all-nighter. She went into class the next day, weary as can be. Her teacher stated that the test is postponed for next week. Jennifer felt bittersweet about it.”

- Small datasets: scaling up Diffusion-LM is an open problem.
## Control Tasks

<table>
<thead>
<tr>
<th>input (Semantic Content)</th>
<th>output text</th>
</tr>
</thead>
<tbody>
<tr>
<td>food : Japanese</td>
<td>Browns Cambridge is good for Japanese food and also children friendly near The Sorrento.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input (Parts-of-speech)</th>
<th>output text</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPN AUX DET ADJ NOUN NOUN VERB ADP DET NOUN ADP DET NOUN PUNCT</td>
<td>Zizzi is a local coffee shop located on the outskirts of the city.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input (Syntax Tree)</th>
<th>output text</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TOP (S (NP (<em>)(</em>)(<em>)) (VP (</em>)(NP (NP (<em>)(</em>))))))</td>
<td>The Twenty Two has great food</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input (Syntax Spans)</th>
<th>output text</th>
</tr>
</thead>
<tbody>
<tr>
<td>(7, 10, VP)</td>
<td>Wildwood pub serves multicultural dishes and is ranked 3 stars</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input (Length)</th>
<th>output text</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Browns Cambridge offers Japanese food located near The Sorrento in the city centre.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input (left context)</th>
<th>input (right context)</th>
<th>output text</th>
</tr>
</thead>
<tbody>
<tr>
<td>My dog loved tennis balls.</td>
<td>My dog had stolen every one and put it under there.</td>
<td>One day, I found all of my lost tennis balls underneath the bed.</td>
</tr>
</tbody>
</table>

- Six controllable generation tasks.
Baselines

• For classifier-guided control:
  ▹ PPLM (Dathathri et al., 2020).
  ▹ FUDGE (Yang and Klein, 2021).
  ▹ Finetuning (skyline).

• For infilling:
  ▹ DELOREAN (Qin et al., 2020).
  ▹ COLD (Qin et al., 2021).
  ▹ Finetuning (skyline).
A Qualitative Example

<table>
<thead>
<tr>
<th>Syntactic Parse</th>
<th>FUDGE</th>
<th>Diffusion-LM</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S (S (NP *)) (VP *(NP *(NP **)) *(VP *(NP *(NP *(ADJP *) **)))))) *(S (NP **)) *(VP *(ADJP *)) ))))))</td>
<td>Zizzi is a cheap restaurant. [incomplete]</td>
<td>Zizzi is a pub providing family friendly Indian food Its customer rating is low Cocum is a Pub serving moderately priced meals and the customer rating is high</td>
<td></td>
</tr>
<tr>
<td>Syntactic Parse</td>
<td>FUDGE</td>
<td>Diffusion-LM</td>
<td>FT</td>
</tr>
<tr>
<td>(S (S (VP *(PP *(NP **)))) *(NP **)) *(VP *(NP *(NP **)) *(SBAR *(WHNP *)) *(S (VP *(NP **)))))) * )</td>
<td>In the city near The Portland Arms is a coffee and fast food place named The Cricketers which is not family - friendly with a customer rating of 5 out of 5.</td>
<td>Located on the Riverside, The Rice Boat is a restaurant that serves Indian food.</td>
<td>Located near The Sorrento, The Mill is a pub that serves Indian cuisine.</td>
</tr>
</tbody>
</table>

- FUDGE and Finetuning (FT) deviate after a few tokens (exposure bias).
- Diffusion-LM is robust to local failures to apply the control.
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Talk Outline

• Constructing Diffusion-LM
  ★ The Standard Diffusion Model (Ho et. al., 2020)
  ★ Learning Word Embeddings (End-to-End Training)
  ★ Predicting the Noiseless Embeddings

• Sampling from Diffusion-LM
  ★ The Clamping Trick and Other Heuristics
  ★ Classifier-Guided Sampling

• Experiments

• Discussion
Limitations

- Decoding is much slower than AR Transformer models: 2000 diffusion steps versus (short) sequence-length AR steps.
- Training seems to converge more slowly than for AR models.
- Diffusion-LM has higher perplexity than comparably-sized AR models.
- Similar challenges for diffusion models of images have been observed and overcome, so there is reason to be hopeful!
Coarse-to-Fine Generation

• Coarse-to-fine generation seems helpful for control (versus autoregressive generation) because the control target can be incorporated globally into the plan for generating text.

• You can use Langevin dynamics to sample coarse-to-fine from AR models fine-tuned in noisy data (Jayaram and Thickstun, 2021)

• A mystery: I spent several months trying to adapt these methods to text, but couldn’t get the sampling to work very well.

• Maybe coarse-to-fine isn’t the only thing that’s going right here?
Non-Autoregressive Language Generation

• Previously non-autoregressive open-ended language generation has seemed difficult (GAN, VAE).

• Adapting the DDPM recipe to text required some alterations, but if similarly-scoped alterations would make VAE’s work well for text it seems like this would have happened by now.

• What (if anything) is different about diffusion models?
Continuous vs. Discrete Models

• NLP folks spend a lot of time trying to convert their discrete data into continuous representations.

• Vision folks spend a lot of time trying to convert their continuous data into discrete representations.

  ▶ VQ-VAE2 (Razavi et al., 2019)
  ▶ VQ-GAN (Esser et al., 2021)
  ▶ Parti (Yu et al., 2022)

• What is going on here?
Thank You!

Code:   https://github.com/XiangLi1999/Diffusion-LM