Diffusion-LM Improves Controllable Text Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Controlling the behavior of language models (LMs) without re-training is a major open problem in natural language generation. While recent works have demonstrated successes on controlling simple sentence attributes (e.g., sentiment), there has been little progress on complex, fine-grained controls (e.g., syntactic structure). To address this challenge, we develop a new *non-autoregressive* language model based on *continuous* diffusions that we call Diffusion-LM. Building upon the recent successes of diffusion models in continuous domains, Diffusion-LM iteratively denoises a sequence of Gaussian vectors into word vectors, yielding a sequence of intermediate latent variables. To control its generation, we iteratively perform gradient updates on these intermediate variables. Diffusion-LM has three properties that enable complex, fine-grained controllable text generation: the continuous nature of diffusion models enables gradient-based control; the non-autoregressive generation order enables more complex, global controls; and incremental denoising induces a coarse-to-fine hierarchy, which facilitates control at multiple granularities. We demonstrate successful control of Diffusion-LM for six challenging fine-grained control tasks, significantly outperforming prior work.

1 Introduction

Large autoregressive language models (LMs) are capable of generating high quality text [30, 3, 5, 40], but in order to reliably deploy these LMs in real world applications, the text generation process needs to be *controllable*: we need to generate text that satisfies desired requirements (e.g. topic, syntactic structure). A natural approach for controlling a LM would be to fine-tune the LM using supervised data of the form (control, text) [15]. However, updating the LM parameters for each control task can be expensive and does not allow for compositions of multiple controls (e.g. generate text that is both positive sentiment *and* non-toxic). This motivates light-weight and modular plug-and-play approaches [6] that keep the LM frozen and steer the generation process using an external predictor that measures how well the generated text satisfies the control. But even then, steering a frozen autoregressive LM has been shown to be difficult, and existing successes have been limited to simple, attribute-level controls (e.g., sentiment or topic) [6, 20, 39].

In order to broaden the set of viable controls, we 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, as shown in Figure 1. These gradual denoising steps produce a coarse-to-fine hierarchy of continuous latent representations.

Diffusion-LM enables new forms of complex, fine-grained control tasks that are not currently possible using autoregressive LMs. We highlight three desirable properties of Diffusion-LM that may enable these capabilities. First, Diffusion-LM directly generates *continuous* latent representations, which can be updated and controlled using gradients derived from external predictors. Second, diffusion LM is a *non-autoregressive* model that generates all tokens in parallel. This allows it to incorporate complex, global controls. As a bonus, it handles infilling at decoding time without additional predictors or

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

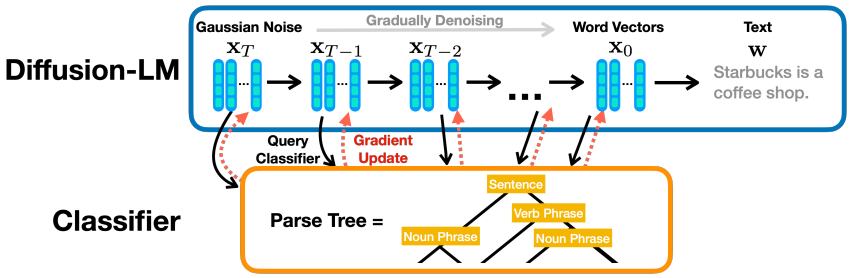

Figure 1: Diffusion-LM iteratively denoises a sequence of Gaussian vectors into word vectors, yielding a intermediate latent variables of decreasing noise level $\mathbf{x}_T \cdots \mathbf{x}_0$. For controllable generation, we iteratively perform gradient updates on these continuous latents to optimize for fluency (parametrized by Diffusion-LM) and satisfy control requirements (parametrized by a predictor).

specialized techniques, unlike autoregressive LMs which require expensive search or marginalization steps [21, 38, 28]. Finally, Diffusion-LM induces a *coarse-to-fine* hierarchy of continuous latent representations, which enable controls that operate on the entire sequence (e.g. sentiment or length) as well as on individual words (e.g. parts of speech).

Continuous diffusion models have been extremely successful in vision and audio domains [11, 19, 31, 7, 4], but they have not been applied to text because of the inherently discrete nature of text (§3). Adapting this class of models to text requires several modifications to the diffusion training objective and decoding procedure (§4). We control Diffusion-LM using a gradient-based method, as shown in Figure 1. This method enables us to steer the text generation process towards outputs that satisfy given structural and semantic control targets. It iteratively performs gradient updates on the continuous latent variables of Diffusion-LM to balance fluency and control satisfaction (§4.3).

To demonstrate control of Diffusion-LM, we consider a variety of control targets ranging from simple attributes (e.g., sentence length) to complex structures (e.g., parse tree) and semantic content. Our method almost doubles the success rate of previous plug-and-play methods and matches or outperforms the fine-tuning oracle on all these predictor-guided control tasks (§6.1). In addition to these individual control tasks, we show that we can successfully compose multiple predictor-guided controls to generate sentences with both desired semantic content and syntactic structure (§6.2). Finally, we also consider span-anchored controls, such as length control and infilling. These tasks are predictor free, and our Diffusion-LM significantly outperforms prior plug-and-play methods and is on-par with an autoregressive LM trained from scratch for the infilling task (§6.3).

2 Related Work

Diffusion Models for Text. Diffusion models [35] have demonstrated great success in continuous data domains [11, 25, 19, 23], producing images and audio that have state-of-the-art sample quality. To handle discrete data, past works have studied text diffusion models on discrete state spaces, which defines a corruption process on discrete data (e.g., each token has some probability to be corrupted to an absorbing or random token) [1, 13, 14]. In this paper, we focus on *continuous* diffusion models for text and to the best of our knowledge, our work is the first to explore this setting. In contrast to discrete diffusion LMs, our continuous diffusion LMs induce continuous latent representations, which enables efficient gradient-based methods for controllable generation.

Autoregressive and Non-autoregressive LMs. Most large pre-trained LMs are left-to-right autoregressive (e.g., GPT-3 [3], PaLM [5]). The fixed generation order limits the models' flexibility in many controllable generation settings, especially those that impose controls on the right contexts. Since autoregressive LMs cannot directly condition on right contexts, prior works have developed specialized training and decoding techniques for these tasks [34, 8, 28]. For example, Qin et al. [29] is a decoding method that relaxes the discrete LM outputs to continuous variables and backpropagates gradient information from the right context. Diffusion-LM can condition on arbitrary predictors that look at complex, global properties of the sentence. There are other non-autoregressive LMs that have been developed for machine translation and speech-to-text tasks [10, 33]. However these methods are specialized for speech and translation settings, where the entropy over valid outputs is low, and whether they work for language modeling remains an open problem. We leave detailed discussions to appendix H.

Plug-and-Play Controllable Generation. Controllable text generation is the task of decoding from a conditional distribution $p(\mathbf{w}|\mathbf{c})$, where $\mathbf{w}$ is the text sequence, and $\mathbf{c}$ is the control constraint. Plug-and-play methods leverage Bayes rule to control the output of an unconditional LM $p(\mathbf{w})$ at decoding time: $p(\mathbf{w}|\mathbf{c}) \propto p(\mathbf{w}) \cdot p(\mathbf{c}|\mathbf{w})$ where $p(\mathbf{w})$ is the frozen LM, and $p(\mathbf{c}|\mathbf{w})$ is a predictor probability of whether a sequence fulfills the goal of the control task. There are several plug-and-play approaches based on autoregressive LMs: FUDGE [39] reweights the LM prediction at each token with an estimate of $p(\mathbf{c}|\mathbf{w})$ for the partial sequence; GeDi [20] and DExperts [22] reweight the LM prediction at each token with a smaller LM finetuned/trained for the control task.

The closest work to ours is PPLM [6], which runs gradient ascent on an autoregressive LM's hidden activations to steer the next token towards higher $p(\mathbf{w})$ and $p(\mathbf{c}|\mathbf{w})$. Because PPLM is based on autoregressive LMs, it can only generate left-to-right, so PPLM cannot repair its past errors. Despite their success on attribute (e.g., topic) controls, we will show these plug-and-play methods for autoregressive LMs fail on more complex control tasks such as controlling syntactic structure and semantic content in §6.1. We demonstrate that Diffusion-LM is capable of plug-and-play controllable generation by applying predictor-guided gradient updates to the continuous sequence of latent variables induced by the Diffusion-LM.

3 Problem Statement and Background

We aim to apply continuous diffusion models to discrete text and begin by defining the problem settings for controllable generation and diffusion modeling.

3.1 Generative Models and Controllable Generation for Text

Consider a language modeling task, where $\mathbf{w} = [w_1 \cdots w_n]$ is a sequence of discrete words drawn from an unknown data distribution. A language model p_{lm} is trained to emulate this data distribution by maximizing the data likelihood: $\mathbb{E}_{\mathbf{w}}[\log p_{\text{lm}}(\mathbf{w})]$. In the controllable generation setting, we have an additional *control* variable $\mathbf{c}$ denoting a feature of interest for $\mathbf{w}$. For syntactic control, $\mathbf{c}$ may be the syntax tree of $\mathbf{w}$ (Figure 1). For sentiment control, $\mathbf{c}$ may be the sentiment label on $\mathbf{w}$. The goal of controllable generation is to approximate samples from the conditional distribution $p(\mathbf{w} \mid \mathbf{c})$.

Controllable generation can be treated as a standard language modeling task using paired data $(\mathbf{w}, \mathbf{c})$. However this approach has two drawbacks: first, tuning p_{lm} from scratch can be computationally expensive; second, there may be a fundamental asymmetry in data collection, where it is substantially easier to collect un-annotated samples $\mathbf{w}$ than paired samples $(\mathbf{w}, \mathbf{c})$. The plug-and play approach seeks to address both concerns by training a large p_{lm} on $\mathbf{w}$ and then steering this model using a lightweight classifier trained on paired data $p(\mathbf{c} \mid \mathbf{w})$. This classifier can guide text generation via Bayes rule as $p(\mathbf{w} \mid \mathbf{c}) \propto p_{\text{lm}}(\mathbf{w}) \cdot p(\mathbf{c} \mid \mathbf{w})$, where $p_{\text{lm}}(\mathbf{w})$ encourages $\mathbf{w}$ to be fluent, and the $p(\mathbf{c} \mid \mathbf{w})$ encourages $\mathbf{w}$ to fulfill the constraints.

3.2 Diffusion Models for Continuous Domains

A diffusion model [11, 25] is a latent variable model that models the data $\mathbf{x}_0 \sim p_{\text{data}}$ as a Markov chain $\mathbf{x}_T \dots \mathbf{x}_0$, where $\mathbf{x}_T$ is a Gaussian, and $\mathbf{x}_{t-1} \mid \mathbf{x}_t$ is a *de-noising* step that gradually transforms noisy intermediate variables into the observed data distribution (Figure 2). This sequence of continuous latent variables $\mathbf{x}_{1:T}$ is defined by a forward process that incrementally adds Gaussian noise to data $\mathbf{x}_0$ until, at diffusion step T, samples $\mathbf{x}_T$ are approximately Gaussian. Each transition $\mathbf{x}_{t-1} \to \mathbf{x}_t$ is parametrized by $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$, where the hyperparameter β_t is the amount of noise added at diffusion step t.

The diffusion model generates samples by reversing this process: it incrementally denoises the sequence of latent variables $\mathbf{x}_{T:1}$ to approximate samples from the target distribution. Each denoising transition $\mathbf{x}_t \to \mathbf{x}_{t-1}$ is parametrized by the model $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$.

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 $\log p_\theta(\mathbf{x}_0)$ [35]:

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

However, this objective can be unstable and require many optimization tricks to stabilize [25]. To circumvent this issue, Ho et al. [11] devised a simple surrogate objective that expands and reweights

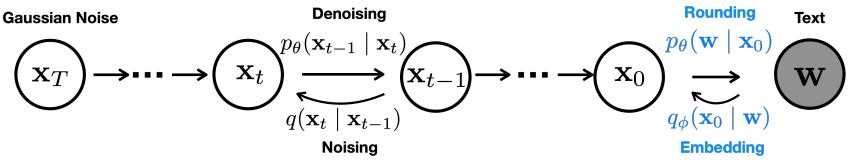

Figure 2: A graphical model representing the forward and reverse diffusion processes. In addition to the original diffusion models [11], we add a Markov transition between $\mathbf{x}_0$ and $\mathbf{w}$, and propose the embedding §4.1 and rounding §4.2 techniques.

each KL-divergence term in $\mathcal{L}_{\text{vlb}}$ to obtain a mean-squared error loss which we will refer to as

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

where $\hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)$ is the mean of the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$. While $\mathcal{L}_{\text{simple}}$ is no longer a valid lower bound, prior work has found that it empirically made training more stable and improved sample quality [1]. We will make use of similar simplifications in Diffusion-LM to stabilize training and improve sample quality (§4.1).

4 Diffusion-LM: Continuous Diffusion Language Modeling

Constructing Diffusion-LM requires several modifications to the standard diffusion model. First, we must define an embedding function that maps discrete text into a continuous space. To address this, we propose an end-to-end training objective for learning embeddings (§4.1). Second, we require a rounding method to map vectors in embedding space back to words. To address this, we propose training and decoding time methods to facilitate rounding (§4.2). The two improvements make it possible to reliably train Diffusion-LMs, and we describe how to perform plug-and-play controllable generation on these models using classifier guidance (§4.3).

4.1 End-to-end Training

To apply a continuous diffusion model to discrete text, we define an embedding function $\text{EMB}(w_i)$ that maps each word to a vector in $\mathbb{R}^d$. We define the embedding of a sequence $\mathbf{w}$ of length n to be: $\text{EMB}(\mathbf{w}) = [\text{EMB}(w_1), \ldots, \text{EMB}(w_n)] \in \mathbb{R}^{nd}$.

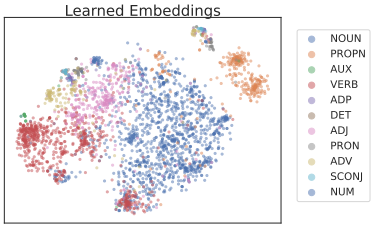

Figure 3: A t-SNE [37] plot of the learned word embeddings.

We propose a modification of the diffusion model training objective (Equation 1) that jointly learns the diffusion model's parameters and word embeddings. In preliminary experiments, we explored random Gaussian embeddings, as well as pre-trained word embeddings [27, 30]. We found that these fixed embeddings are suboptimal for Diffusion-LM compared to end-to-end training[2].

As shown in Figure 2, our approach adds a Markov transition from discrete words $\mathbf{w}$ to $\mathbf{x}_0$ in the forward process, parametrized by $q_\phi(\mathbf{x}_0|\mathbf{w}) = \mathcal{N}(\text{EMB}(\mathbf{w}), \sigma_0 I)$. In the reverse process, we add a trainable rounding step, parametrized by $p_\theta(\mathbf{w} \mid \mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i \mid x_i)$, where $p_\theta(w_i \mid x_i)$ is a softmax distribution. The training objectives introduced in §3 now becomes

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

We derive $\mathcal{L}^{\text{e2e}}_{\text{simple}}(\mathbf{w})$ from $\mathcal{L}^{\text{e2e}}_{\text{vlb}}(\mathbf{w})$ following the simplification in §3.2 and our derivation details are shown in Appendix D. Since we are training the embedding function, q_ϕ now contains trainable

[1] Our definition of $\mathcal{L}_{\text{simple}}$ uses a different parametrization from Ho et al. [11]. We define our squared loss in terms of $\mu_\theta(\mathbf{x}_t, t)$ while they express it in terms of $\epsilon_\theta(\mathbf{x}_t, t)$.

[2] While trainable embeddings perform best on control and generation tasks, we found that fixed embeddings onto the vocabulary simplex were helpful when optimizing for held-out perplexity. We leave discussion of this approach and perplexity results to Appendix C as the focus of this work is generation quality and not perplexity.

parameters and we use the reparametrization trick [32, 17] to backpropagate through this sampling step. Empirically, we find the learned embeddings cluster meaningfully: words with the same part-of-speech tags (syntactic role) tend to be clustered, as shown in Figure 3.

4.2 Reducing Rounding Errors

The major challenge in applying diffusion models to text is mapping between discrete text ($\mathbf{w}$) and continuous latent variables $\mathbf{x}_0$. The learned embeddings in §4.2 define an embedding that maps discrete texts to our continuous space. We now describe the inverse process of rounding a continuous latent variable $\mathbf{x}_0$ into discrete text.

Ideally, the denoising process itself should learn that the distribution over $\mathbf{x}_0$ is nearly a mixture of low-variance distributions (where each mixture component represents a word). In this case, the rounding step would be unambiguous. However, we found that diffusion models do not seem to learn this mixture structure of $\mathbf{x}_0$.

One explanation for this phenomenon is that our objective $\mathcal{L}_{\text{simple}}$ puts insufficient emphasis on modeling the mixture structure of $\mathbf{x}_0$. Recall that we defined $\mathcal{L}_{\text{simple}} = \sum_{t=1}^{T} ||\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)||^2$, where our model predicts individual denoising steps $\mathbf{x}_{t-1} \mid \mathbf{x}_t$. In this objective, the constraint that $\mathbf{x}_0$ is (nearly) a mixture of Dirac delta distribution will only appear in the terms with t near zero, and we found that this parametrization required careful tuning to force the objective to emphasize those terms (see Appendix B).

Our approach is to re-parametrize $\mathcal{L}_{\text{simple}}$ to force the model to explicitly model $\mathbf{x}_0$ in *every* term of the objective. Specifically, we select an alternative parametrization $\mathcal{L}_{\text{simple}} = \sum_{t=1}^{T} ||\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_0||^2$, where our model $\mathbf{x}_\theta(\mathbf{x}_t, t)$ predicts $\mathbf{x}_0$ directly [3]. This forces the neural network to predict $\mathbf{x}_0$ in every term and we found that models trained with this objective quickly learn the mixture structure.

We described how re-parametrization can be helpful for model training, but we also found that the same idea could be used when generating from the model in a technique that we call the *clamping* trick. In the standard generation approach, the model denoises $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$ by first computing an estimate of $\mathbf{x}_0$ via $\mathbf{x}_\theta(\mathbf{x}_t, t)$ and then sampling $\mathbf{x}_{t-1}$ conditioned on this estimate: $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}}\mathbf{x}_\theta(\mathbf{x}_t, t) + \sqrt{1-\bar{\alpha}}\epsilon$, where $\bar{\alpha}_t = \prod_{s=0}^{t}(1-\beta_s)$ and $\epsilon \sim \mathcal{N}(0, I)$ [4]. In the clamping trick, the model additionally maps the predicted vector $\mathbf{x}_\theta(\mathbf{x}_t, t)$ to its nearest word embedding sequence. Now, the sampling step becomes $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}} \cdot \text{Clamp}(\mathbf{x}_\theta(\mathbf{x}_t, t)) + \sqrt{1-\bar{\alpha}}\epsilon$. The clamping trick forces the predicted embedding to commit to a word for intermediate diffusion steps, making the vector predictions more precise and reducing rounding errors.

4.3 Controllable Text Generation

With the above improvements, we are able to train Diffusion-LMs that generate fluent text. We now describe a procedure that enables plug-and-play control on this Diffusion-LM. Our approach to control is inspired by the Bayesian formulation in §3.1, but instead of performing control directly on the discrete text, we perform control on the sequence of continuous latents $\mathbf{x}_{0:T}$ defined by Diffusion-LM, and apply the rounding step to convert these latents into text.

Controlling $\mathbf{x}_{0:T}$ is equivalent to decoding from the posterior $p(\mathbf{x}_{0:T}|\mathbf{c}) = \prod_{t=1}^{T} p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$, and we decompose this joint inference problem to a sequence of control problems at each diffusion 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 [36], leading to:

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

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.

[3] Predicting $\mathbf{x}_0$ and $\mathbf{x}_{t-1}$ is equivalent up to scaling constants as the distribution of $\mathbf{x}_{t-1}$ can be obtained in closed form via the forward process $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}}\epsilon$, see Appendix D for further details.

[4] This follows from the marginal distribution $q(\mathbf{x}_t \mid \mathbf{x}_0)$, which is a closed form Gaussian since all the Markov transitions are Gaussian.

Similar to work in the image setting [7, 36], we train the classifier on the diffusion latent variables and run gradient updates on the latent space $\mathbf{x}_{t-1}$ to steer it towards fulfilling the control. To improve performance on text and speed up decoding, we introduce two key modifications.

To improve decoding speed, we downsample the diffusion steps from 2000 to 200. For each downsampled time step, we run 3 steps of the Adagrad[5] [9] update on $\lambda \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) + \log p(\mathbf{c} \mid \mathbf{x}_{t-1})$, where λ is a hyperparameter that trades off fluency (the first term) and control (the second term). While existing controllable generation methods for diffusions do not include the $\lambda p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ term in the objective, we found this term to be instrumental for generating fluent text. The resulting controllable generation process can be viewed as a stochastic decoding method that balances maximizing and sampling $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$, much like popular text generation techniques like nucleus sampling [12].

5 Experimental Setup

5.1 Datasets and Hyperparameters

We train Diffusion-LM on two datasets: E2E [26] and ROCStories [24]. 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. This dataset is more challenging to model than E2E, because the stories contain a larger vocabulary of 11K words and more diverse semantic content.

Our diffusion model consists of 80M parameters, the noise schedule (β_t) to be square-root, with a sequence length of 64 and $2k$ diffusion steps. We treat the embedding dimension as a hyperparameter, setting $d = 16$ for E2E and $d = 128$ for ROCStories. See the appendix for all other training details. At decoding time, we downsample to 200 diffusion steps for E2E and maintain 2000 steps for ROCStories. Admittedly, decoding Diffusion-LM is still slower than decoding autoregressive LMs.

5.2 Control tasks

We consider 6 control tasks: the first 4 tasks rely on an external predictor, and the last 2 tasks are predictor free[6]. For each control task (e.g. semantic content), we sample 200 control targets $\mathbf{c}$ (e.g., rating=5 star) from the validation splits, and we generate 50 samples for each control target. To evaluate the fluency of the generated text, we feed them to a teacher LM (i.e., a carefully fine-tuned GPT-2 model) and report the perplexity of generated text under the teacher LM. We call this metric lm-score (denoted as lm): a lower lm-score indicates better sample quality. We define success metrics for each control task.

Semantic Content. Given a field (e.g., rating) and value (e.g., 5 star), we aim to generate a sentence that covers field=value, and report success rate by exact match of 'value'.

Parts-of-speech. Given a sequence of parts-of-speech (POS) tags (e.g., *Pronoun Verb Determiner Noun*), we aim to generate a sequence of words of the same length whose POS tags (under an oracle POS tagger) match the target (e.g., *I ate an apple*). We quantify success via word-level exact match.

Syntax Tree. Given a target syntax tree (see Figure 1), we aim to generate text with a matching syntax tree (under an off-the-shelf parser [18]), quantifying success with F1 scores.

Syntax Span. Instead of controlling the entire tree, our goal is to generate text whose oracle constituent from positions i to j matches a target label (e.g. prepositional phrase). We quantify success via the fraction of spans that match exactly.

Length. Given a target length $10, \ldots, 40$, our goal is to generate a sequence with a length within ± 2 of the target. In the case of Diffusion-LM, we treat this as a classifier-free control task.

Infilling. Given a left context (O_1) and a right context (O_2) from the aNLG dataset [2], and the goal is to generate a sentence that logically connects O_1 and O_2. For evaluation, we report both automatic and human evaluation from the Genie leaderboard [16].

[5] We tried ablations that replaced Adagrad with SGD, but we found Adagrad to be substantially less sensitive to hyperparameter tuning.

[6] Length is predictor free for our Diffusion-LM based methods, but other baselines still require a predictor.

Table 1: 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 trees and spans.

	Semantic		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

5.3 Predictor-Guided Control Baselines

For the first 5 control tasks, we compare our method with PPLM, FUDGE, and a fine-tuning oracle. Both PPLM and FUDGE are plug-and-play controllable generation approaches based on an autoregressive LM, which we train from scratch using the GPT-2 small [30].

PPLM[6]. This method runs gradient ascent on the LM activations to increase the classifier probabilities and language model probabilities, and has been successful on simple attribute control. We apply PPLM to control semantic content, but not the remaining 4 tasks which require positional information, as PPLM's classifier lacks positional information.

FUDGE[39]. For each control task, FUDGE requires a future discriminator that takes in a prefix sequence and predicts whether the complete sequence would satisfy the constraint. At decoding time, FUDGE reweights the LM prediction by the discriminator scores.

FT. For each control task, we fine-tune GPT-2 on (control, text) pair. We report both the sampling and beam search outputs of the fine-tuned models, denoted as FT-sample and FT-search, respectively. Note that this is an oracle, since it requires fine-tuning the LM parameters.

5.4 Infilling Baselines

We compare to 3 specialized baseline methods developed in past work for the infilling task.

DELOREAN [28]. This method continuously relaxes the output space of a left-to-right autoregressive LM, and iteratively performs gradient updates on the continuous space to enforce fluent connection to the right contexts. This yields a continuous vector which is rounded back to text.

COLD[29]. COLD specifies an energy-based model that includes fluency (from left-to-right and right-to-left LM) and coherence constraints (from lexical overlap). It samples continuous vectors from this energy-based model and round them to text.

AR-infilling. We train an autoregressive LM from scratch to do sentence infilling task [8]. Similar to training Diffusion-LM, we train on the ROCStories dataset, but pre-process it by reordering sentences from $(O_1, O_{\text{middle}}, O_2)$ to $(O_1, O_2, O_{\text{middle}})$. At evaluation time, we feed in O_1, O_2, and the model generates the middle sentence.

6 Results

We train Diffusion-LMs on the E2E and ROCStories datasets, and compare to baseline autoregressive models (GPT-2) with comparable parameter counts. Diffusion-LM has worse holdout log-likelihood[7] than a comparably sized GPT-2 model for both datasets (E2E: 2.28 v.s. 1.77 , ROCStories: 3.88 v.s 3.05) although we begin to bridge this gap by doubling the size of our Diffusion-LM and training on more data (ROCStories: 3.10 v.s. 3.05). Despite the lower perplexity, controllable generation based on our Diffusion-LM results in significantly better outputs than systems based on autoregressive LMs.

6.1 Predictor-Guided Controllable Text Generation Results

As shown in Table 1, Diffusion-LM achieves high success and fluency across all predictor-guided control tasks. It significantly outperforms the PPLM and FUDGE baselines across all 5 tasks. Surprisingly, our method outperforms the fine-tuning oracle on the syntax tree control and the span control tasks and achieves similar performance on the remaining 3 tasks.

[7] Exact log-likelihoods are intractable for Diffusion-LM, so we report the lower bound $\mathcal{L}_{\text{vlb}}^{\text{e2e}}$.

Table 2: Qualitative examples from the syntax tree control tasks. The target parse is linearized by nested brackets representing the constituents: S is sentence, NP is noun phrase, VP is verb phrase, PP is prepositional phrase, etc. Tokens within each span are represented as * . We color failing spans red and **bold** the spans of interest that we discuss in the text.

Target parse	(S (S (NP *) (VP * (NP (NP * *) (VP * **(NP (ADJP * *) *)**))))) * (S (NP * * *) (VP * (ADJP (ADJP *)))))
FUDGE	Zizzi is a cheap restaurant . **[incomplete]**
Diffusion-LM	Zizzi is a pub providing **family friendly Indian food** Its customer rating is low
FT	Cocum is a Pub serving **moderately priced meals** and the customer rating is high
Target parse	(S (S (VP * (PP * (NP * *))))) * **(NP * * *)** (VP * (NP (NP * *) (SBAR (WHNP *) (S (VP * (NP * *)))))) *)
FUDGE	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 .
Diffusion-LM	Located on the riverside , **The Rice Boat** is a restaurant that serves Indian food .
FT	Located near The Sorrento, **The Mill** is a pub that serves Indian cuisine.

Table 3: In this experiment, we compose semantic control and syntactic control: Diffusion-LM achieves good 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. syntax tree and POS).

	Semantic + Syntax Tree			Semantic + POS		
	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**

Controlling the syntax tree and spans are challenging tasks for fine-tuning, because conditioning on the syntax tree requires reasoning about the nested structure of the parse tree, and conditioning on spans requires lookahead planning to ensure the right constituent appears at the target position.

We observe that PPLM fails in the semantic content control task and conjecture that this is because PPLM is designed to control coarse-grained attributes, and may not be useful for more targeted tasks such as enforcing that a restaurant review contains a reference to Starbucks.

FUDGE performs well on semantic content control but does not perform well on the remaining four tasks. Controlling a structured output (POS and syntax tree) is hard for FUDGE because making one mistake anywhere in the prefix makes the discriminator assign low probabilities to all continuations. In other control tasks requiring planning (Length and Spans), the future discriminator is difficult to train, as it must implicitly perform lookahead planning.

The non-autoregressive nature of our Diffusion-LM allows it to easily solve all the tasks that require precise planning (spans and length). We believe that it works well for complex controls that involve global structures (POS, syntax parse) because the coarse-to-fine representations allow the predictors to exert control on the entire sequence (near $t = T$) as well as on individual tokens (near $t = 0$).

Qualitative Results. Table 2 shows samples of syntax tree control. Our method and fine-tuning both provide fluent sentences that mostly satisfy controls, whereas FUDGE deviates from the constraints after the first few words. One key difference between our method and fine-tuning is that Diffusion-LM is able to correct for a failed span and have suffix spans match the target. In the "Family friendly Indian food" example, the span is wrong because the generated span contains 1 more word than the target. Fortunately, this error doesn't propagate to later spans, since the model adjusts by dropping the conjunction. Analogously, in the "The Mill" example, FT model generates a failed span, but it fails to adjust it in the suffix, leading to many mis-aligned errors in the suffix.

6.2 Composition of Controls

One unique capability of plug-and-play controllable generation is its modularity. Given predictors for multiple independent tasks, gradient guided control makes it simple to generate from the intersection of multiple controls by taking gradients on the sum of the predictor log-probabilities.

We evaluate this setting on the combination of semantic content + syntax tree control and semantic content + POS tag control. As shown in Table 3, our Diffusion-LM achieves a high success rate for

Table 4: For sentence infilling, Diffusion-LM significantly outperforms prior work COLD [29] and Delorean [28] (numbers taken from paper), and matches the performance of an autoregressive LM (AR) trained from scratch to do infilling.

	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**	$\mathbf{0.39}^{+0.02}_{-0.03}$

both of the two components, whereas FUDGE gives up on the more global syntactic control. This is expected because FUDGE fails to control syntax on its own.

Fine-tuned models are good at POS and semantic attribute control individually but do not compose these two controls well by product of experts (PoE), leading to a large drop in success rates for both constraints.

6.3 Infilling Results

As shown in Table 4, our diffusion LM significantly outperforms continuous relaxation based methods for infilling (COLD and Delorean). Moreover, our method achieves comparable performance to fine-tuning a specialized model for this task. Our method has slightly better automatic evaluation scores and the human evaluation found no statistically significant improvement for either method. These results suggest that Diffusion LM can solve many types of controllable generation tasks that depend on generation order or lexical constraints (such as infilling) without specialized training.

6.4 Ablation Studies

We verify the importance of our proposed design choices in §4 through two ablation studies. We measure the sample quality of Diffusion-LM using the lm-score on 500 samples §5.2.

Learned v.s. Random Embeddings (§4.1). Learned embeddings outperform random embeddings on the ROCStories, which is a harder language modeling task. The same trend holds for the E2E dataset but with a smaller margin.

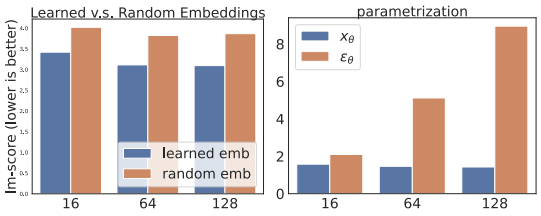

Figure 4: We measure the impact of our proposed design choices through the lm-score. We find both learned embeddings and reparametrization substantially improves sample quality.

Objective Parametrization (§4.2). We propose to let the diffusion model predict $\mathbf{x}_0$ directly. Here, we compare this with standard parametrization in image generation which parametrizes by the noise term ϵ. Figure 4 (right) shows that parametrizing by $\mathbf{x}_0$ consistently attains good performance across dimensions, whereas parametrizing by ϵ works fine for small dimensions, but quickly collapses for larger dimensions.

7 Conclusion and Limitations

We proposed Diffusion-LM, a novel and controllable language model based on continuous diffusions, which enables new forms of complex fine-grained control tasks. We demonstrate Diffusion-LM's success in 6 fine-grained control tasks: our method almost doubles the control success rate of prior methods, and is competitive with baseline fine-tuning methods that require additional training.

Admittedly, Diffusion-LM has some drawbacks relative to autoregressive LMs: (1) it suffers from higher perplexity; (2) decoding is substantially slower; and (3) training converges more slowly. Despite these limitations, we find the degree of control enabled by Diffusion-LM compelling. We hope that the ability to control Diffusion-LM that we have demonstrated will motivate further work to refine and scale this language modeling technique, overcoming its current limitations.

References

[1] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=h7-XixPCAL.

[2] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. Abductive commonsense reasoning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Byg1v1HKDB.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[4] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NsMLjcFaO8O.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. 2022.

[6] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

[7] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

[8] Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.225. URL https://aclanthology.org/2020.acl-main.225.

[9] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *J. Mach. Learn. Res.*, 2010.

[10] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851, 2020.

[12] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

[13] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *arXiv preprint arXiv:2102.05379*, 2021.

[14] Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Lm8T39vLDTE.

[15] N. Keskar, B. McCann, L. R. Varshney, Caiming Xiong, and R. Socher. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858, 2019.

[16] Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A. Smith, and Daniel S. Weld. Genie: A leaderboard for human-in-the-loop evaluation of text generation. *ArXiv*, abs/2101.06561, 2021.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[18] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1249. URL https://aclanthology.org/P18-1249.

[19] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[20] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. GeDi: Generative Discriminator Guided Sequence Generation. *arXiv preprint arXiv:2009.06367*, 2020.

[21] Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. Limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5147–5173, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.405. URL https://aclanthology.org/2021.naacl-main.405.

[22] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.522. URL https://aclanthology.org/2021.acl-long.522.

[23] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, March 2021.

[24] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL https://aclanthology.org/N16-1098.

[25] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.

[26] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5525. URL https://aclanthology.org/W17-5525.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

[28] Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.58. URL https://aclanthology.org/2020.emnlp-main.58.

[29] Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics, 2022. URL https://arxiv.org/abs/2202.11705.

[30] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[31] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, April 2022.

[32] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[33] Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, 2020.

[34] Lei Sha. Gradient-guided unsupervised lexically constrained text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8692–8703, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.701. URL https://aclanthology.org/2020.emnlp-main.701.

[35] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[37] Laurens van der Maaten and Geoffrey Hinton.

[38] Rose E Wang, Esin Durmus, Noah Goodman, and Tatsunori Hashimoto. Language modeling via stochastic processes. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=pMQwKL1yctf.

[39] Kevin Yang and Dan Klein. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.276. URL https://aclanthology.org/2021.naacl-main.276.

[40] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] in the conclusion and limitation section also in the first paragraph of result section.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]