SSD-LM: Semi-autoregressive Simplex-based Diffusion Language Model for Text Generation and Modular Control

Xiaochuang Han[♠] **Sachin Kumar**[♣] **Yulia Tsvetkov**[♠]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]Language Technologies Institute, Carnegie Mellon University
{xhan77, yuliats}@cs.washington.edu[♠] sachink@cs.cmu.edu[♣]

Abstract

Despite the growing success of diffusion models in continuous-valued domains (e.g., images), similar efforts for discrete domains such as text have yet to match the performance of autoregressive language models. In this work, we present SSD-LM—a diffusion-based language model with two key design choices. First, SSD-LM is *semi-autoregressive*, iteratively generating blocks of text, allowing for flexible output length at decoding time while enabling local bidirectional context updates. Second, it is *simplex-based*, performing diffusion on the natural vocabulary space rather than a learned latent space, allowing us to incorporate classifier guidance and modular control using off-the-shelf classifiers without any adaptation. We evaluate SSD-LM on unconstrained text generation benchmarks, and show that it matches or outperforms strong autoregressive GPT-2 models across standard quality and diversity metrics, while vastly outperforming diffusion-based baselines. On controlled text generation, SSD-LM also outperforms competitive baselines, with an extra advantage in modularity.[1]

1 Introduction

Diffusion models (Sohl-Dickstein et al., 2015), trained to iteratively refine noised inputs, have recently emerged as powerful tools for generative modeling in several continuous-valued domains such as images (Ho et al., 2020), audio (Kong et al., 2021), video (Ho et al., 2022), among others. Attempts to adapt them for discrete domains such as text data, however, have only had limited success: prior work have shown to be promising on specialized cases and small datasets (Hoogeboom et al., 2021; Austin et al., 2021; Li et al., 2022; Chen et al., 2022), but diffusion models for text still underperform (and thus are not widely adopted) compared to autoregressive language models (AR-LMs) which remain the state-of-the-art general purpose text generators (Radford et al., 2019; Brown et al., 2020).

Despite potential advantages of diffusion models for text, there are two key challenges. First, diffusion models generate text non-autoregressively, i.e., they generate (and update) the entire sequence simultaneously rather than token by token left-to-right. Although this property is useful in practice since each output token is informed by a broader bi-directional context (Lee et al., 2018; Ghazvininejad et al., 2019), it requires pre-defining an output sequence length. This limits the flexibility and applicability of trained models. On the other hand, non-autoregressive training with long sequences is expensive and difficult to optimize. In this work, we propose a *semi-autoregressive* solution which strikes a balance between length flexibility and the ability to alter previously generated tokens.

A major advantage of diffusion models over the current standard of autoregressive LMs is their post-hoc controllability using guidance from auxiliary models such as style classifiers (Dhariwal and Nichol, 2021). However, controllability is hard to achieve without compromises in modularity in diffusion-based LMs for text. To enable diffusion generation into discrete text rather than continuous modalities, prior approaches have employed different approximations, e.g., training with embeddings, character, or byte-level methods (Li et al., 2022; Hoogeboom et al., 2021; Austin et al., 2021; Chen et al., 2022). In contrast, existing mainstream LMs and the guidance classifiers they derive often operate at a sub-word level with sub-word representations trained jointly with the language model (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020). Subsequently, changing the input representations to characters or embeddings requires developing guidance models from scratch, which can be expensive or infeasible in many cases. In this work, we propose a *simplex-based* solution which enables the diffusion over discrete texts while maintaining

[1]Our code and models can be found at https://github.com/xhan77/ssd-lm.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 11575–11596

the advantages of diffusion models with plug-and-control guidance models.

In sum, to enable diffusion-based LMs for text we present SSD-LM (§3), addressing the above two challenges. SSD-LM is trained to generate text semi-autoregressively—generating blocks of tokens left-to-right with bidirectional context within the block—which offers the benefits of both AR-LMs and diffusion models. It supports training with and generating variable-length sequences. At the same time, it allows refinement within the token block, in contrast to token-level autoregressive decoding where previously generated tokens cannot be modified at all. SSD-LM uses the same tokenization as popular AR-LMs, representing discrete text via a distribution (or simplex) defined over the vocabulary and is trained to reconstruct texts from noisy versions of the distributions. Due to its underlying representation, our method also offers an easy and modular way of guided (controlled) generation using off-the-shelf text classifiers under the minimal assumption of shared tokenizer.

Our evaluation experiments show, for the first time, that a diffusion-based LM matches or outperforms strong AR-LMs on standard text generation benchmarks (§4). We evaluate SSD-LM on two tasks: (1) unconstrained prompt-based generation substantially outperforming existing diffusion LM approaches and performing on par with or outperforming strong autoregressive LM GPT-2 (Radford et al., 2019) on both quality and diversity (§4.2); and (2) controlled text generation with guidance from off-the-shelf classifiers (no post-hoc training/adaptation) outperforming competitive controlled text generation baselines (§4.3).

2 Background

2.1 Diffusion model

Since their inception as image generators, diffusion models (and their cousins score-based models (Song and Ermon, 2019)) have been widely adopted as high-quality generative models for multiple data modalities. Here, we briefly describe a simplified view of a canonical method, denoising diffusion probabilistic models (Ho et al., 2020, DDPM) which we adapt in this work for text generation. We assume a given dataset $\mathcal{D} = \{{}^1\boldsymbol{x}_0, \ldots, {}^N\boldsymbol{x}_0\}$ of continuous valued items ${}^i\boldsymbol{x}_0$ (e.g., pixel values of an image) henceforth referred to as $\boldsymbol{x}_0$ for simplicity.

Training Training a diffusion model first involves adding a series of Gaussian noise to the original data $\boldsymbol{x}_0$, through T timesteps:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t \quad (1)$$

where $t \in (1, T)$ and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. $\bar{\alpha}_t = \prod_{t'=1}^{t} \alpha_{t'}$, where $\alpha_{t'}$ follow a predefined schedule such that $\bar{\alpha}_t \to 0$ as $t \to T$. This process is called *forward diffusion*. A diffusion model (parameterized by θ) is trained to reverse this forward process by predicting the added noise $\boldsymbol{\epsilon}_t$ given $\boldsymbol{x}_t$ with the following loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(1,T)}\|\epsilon_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2 \quad (2)$$

Inference To get an output from this model, we sample $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively reconstruct a sample $\boldsymbol{x}_0$ by going back in time,

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)) \quad (3)$$

for $t = T, \ldots, 1$.[2] The key obstacle in using vanilla diffusion models directly as text generators is that language consists of discrete tokens, i.e., a non-continuous $\boldsymbol{x}_0$ to which a continuous valued Gaussian noise cannot be added. We propose a straightforward and effective solution by treating tokens as continuous valued simplexes over the vocabulary (Hoang et al., 2017). Other existing methods addressing this problem are discussed in §5.

2.2 Autoregressive LM

An autoregressive LM model optimizes for the likelihood of a sequence of tokens $w^0, \ldots, w^{L-1}$.

$$p_\theta(\boldsymbol{w}^{0:L}) = \prod_{c=0}^{L-1} p_\theta(w^c \mid \boldsymbol{w}^{<c}) \quad (4)$$

To decode from AR-LMs, one can provide a context $\boldsymbol{w}^{<c}$ and decode the next token w^c iteratively by predicting $p_\theta(w^c \mid \boldsymbol{w}^{<c})$ and sampling from it to get the discrete token (Fan et al., 2018; Holtzman et al., 2020). Prior work has shown that these decoding approaches (and by extension the LMs themselves) are prone to degrade when generating long sequences and often devolve into repeating subsequences (Holtzman et al., 2020; Meister et al., 2022). In addition, such LMs do not provide a natural way to incorporate sequence-level control as tokens are generated one at a time without the ability

[2] We omit an additional noise term z here for simplicity, which is present in DDPM but not in another variant DDIM (Song et al., 2021).

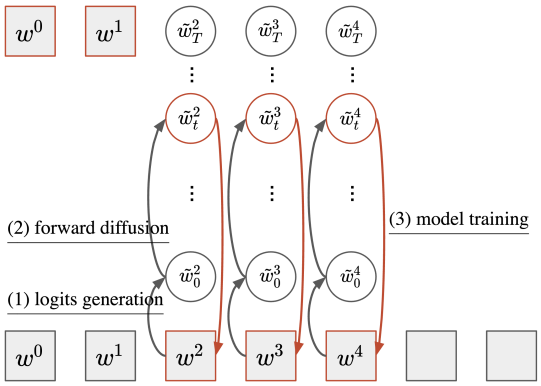

Figure 1: Training SSD-LM (a snapshot at context size $c = 2$, block size $B = 3$). Horizontal axis represents the order of tokens. Vertical axis represents the diffusion timesteps. Shade means observable variables. Square means discrete vocabulary, while circle means continuous logits. Red components are inputs to the learning model θ.

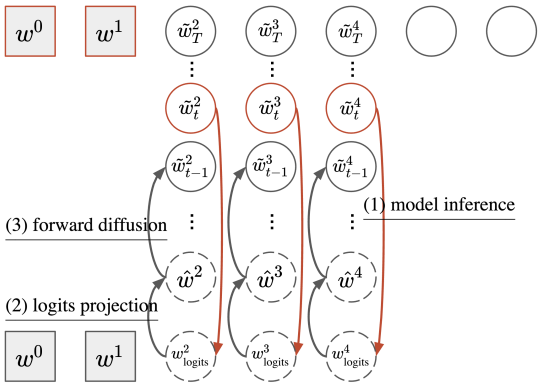

Figure 2: Decoding from SSD-LM (continuing Figure 1). Red components are inputs to the learned model θ. Dash means intermediate variables.

to modify previously generated tokens (Dathathri et al., 2020; Kumar et al., 2022b). In this work, we present a method to train a semi-autoregressive LM that decodes blocks of B tokens at a time, alleviating said issues with the support of diffusion models. Existing literature addressing the two issues individually are discussed in §5.

3 SSD-LM

We introduce SSD-LM—**S**emi-autoregressive **S**implex-based **D**iffusion Language Model— adapting key components from both autoregressive LM and vanilla diffusion models. Conceptually, SSD-LM uses diffusion model to decode $\boldsymbol{w}^{c:c+B}$, a block of tokens of length B, given a Gaussian noise and a context $\boldsymbol{w}^{<c}$ of length c. We show an intuitive diagram and pseudo-code for the training and decoding algorithm of SSD-LM in Figure 1, Figure 2, and Figure 3.

3.1 Training

Continuous data representation To build a continuous representation for discrete tokens, we adopt an *almost-one-hot* simplex representation over the model's vocabulary V. We define a simple operation logits-generation$(.)$ to map a token w to $\tilde{\boldsymbol{w}} \in \{-K, +K\}^{|V|}$ as follows.

$$\tilde{w}_{(i)} = \begin{cases} +K \text{ when } w = V_{(i)} \\ -K \text{ when } w \neq V_{(i)} \end{cases} \quad (5)$$

where i is the index of the vocabulary. We call $\tilde{\boldsymbol{w}}$ the logits for token w, and $\text{softmax}(\tilde{\boldsymbol{w}})$ gives a probability simplex over the vocabulary V, with a probability mass concentrated on the token w. There is no learnable parameter in this mapping.

Forward diffusion Following Ho et al. (2020), we add a time-dependent Gaussian noise to the logits.

$$\tilde{\boldsymbol{w}}_0^{c:c+B} = \text{logits-generation}(\boldsymbol{w}^{c:c+B}) \quad (6)$$

$$\tilde{\boldsymbol{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\boldsymbol{w}}_0^{c:c+B} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t \quad (7)$$

where $t \in (1, T)$, $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$, and $\bar{\alpha}_t \to 0$ as $t \to T$. At the final step T, $\text{softmax}(\tilde{\boldsymbol{w}}_T^{c:c+B})$ are fully noisy simplexes over V, with a logit-normal distribution (Atchison and Shen, 1980).

Loss function In Eq. 2, a diffusion model is trained to predict the added noise from the noisy representations. Since the forward diffusion process can be computed in a single step (Eq. 1), the notion here is equivalent to predicting the original data representation (Song et al., 2021; Li et al., 2022). Our objective follows the same intuition but estimates a likelihood instead of the L2 distance while conditioning on additional context:[3]

$$\mathcal{L}(\theta) = \mathbb{E}[-\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})] \quad (8)$$

$$= \mathbb{E}\left[\sum_{j=c}^{c+B-1} -\log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})\right] \quad (9)$$

$\mathbb{E}[\cdot]$ is a shorthand for $\mathbb{E}_{c\sim\mathcal{U}(1,L-B),t\sim\mathcal{U}(1,T)}[\cdot]$. The architecture for θ throughout this work is a bi-directional Transformer encoder (Vaswani et al., 2017). Specifically, the input to the model is a concatenation of the context $\boldsymbol{w}^{<c}$ and a sequence of noisy vocabulary simplexes $\text{softmax}(\tilde{\boldsymbol{w}}_t^{c:c+B})$ of

[3] L2 distance did not work in our pilot study potentially due to the intrinsically skewed simplex representation.

Algorithm 1 Training

repeat
  $\boldsymbol{w}^{0:L} \sim q(\boldsymbol{w}^{0:L})$
  $c \sim \text{Uniform}(\{1, \ldots, L-B\})$
  $\tilde{\boldsymbol{w}}_0^{c:c+B} = \text{logits-generation}(\boldsymbol{w}^{c:c+B})$
  $t \sim \text{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
  $\tilde{\boldsymbol{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\boldsymbol{w}}_0^{c:c+B} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
  Take gradient descent step on
    $\nabla_\theta[-\sum_{j=c}^{c+B-1} \log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})]$
until converged

Algorithm 2 Decoding (at a given c)

$\tilde{\boldsymbol{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\boldsymbol{w}_{\text{logits}}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$
  $\hat{\boldsymbol{w}}^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits}}^{c:c+B})$ if uncontrolled, else $\hat{\boldsymbol{w}}^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits}}^{c:c+B} + \lambda\nabla_{\boldsymbol{w}} f_\phi(\cdot))$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
  $\tilde{\boldsymbol{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{w}}^{c:c+B} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{z}$
end for
return $\text{argmax}\, \tilde{\boldsymbol{w}}_0^{c:c+B}$

Figure 3: Training and decoding algorithms for SSD-LM. The training algorithm starts with sampling a sequence from the pretraining data $q(\boldsymbol{w}^{0:L})$. The decoding algorithm can be applied m iterations to obtain a $m \cdot B$-token generation, with the returned B tokens at each iteration appended to the previous generation, increasing c.

length B. The target output is the original tokens $\boldsymbol{w}^{c:c+B}$ at positions c to $c+B$.

One minimal modification made to the Transformer model is that in addition to the conventional embedding lookup for $\boldsymbol{w}^{<c}$, we modify the embedding layer to take as input a distribution over the vocabulary, $\text{softmax}(\tilde{\boldsymbol{w}}_t^{c:c+B})$, and compute the embedding vector as a weighted sum of the embedding table. A timestep embedding is also added before the first Transformer block to inform the model of the current timestep.[4]

In §A, we present another interpretation of the training objective as an intuitive contrastive loss.

3.2 Decoding

Logits projection Similar to continuous-valued diffusion models, sampling from SSD-LM involves reverse diffusion from $t = T, \ldots, 1$ starting with a Gaussian noise. At any timestep t, our model θ takes as input noised logits $\tilde{\boldsymbol{w}}_t^{c:c+B}$ and estimates the probability distribution of the original tokens in data by first predicting the logits:

$$\boldsymbol{w}_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c}) \quad (10)$$

which are then converted to a distribution via softmax. To feed this output to the next step of reverse diffusion, $t-1$, we define a logits-projection operation to build a predicted data representation close to the initial data representation (almost-one-hot mapping; Eq. 5). We consider three projection operations.

- *Greedy:* creates an almost-one-hot logit centered at the highest probability token.[5]

$$\hat{w}_{(i)} = \begin{cases} +K \text{ if } i = \text{argmax}(\boldsymbol{w}_{\text{logits}}) \\ -K \text{ otherwise} \end{cases} \quad (11)$$

- *Sampling:* creates an almost-one-hot logit centered around a token sampled from the output distribution using top-p sampling (Holtzman et al., 2020). p is a hyperparameter.

$$\hat{w}_{(i)} = \begin{cases} +K \text{ if } i = \text{top-}p\text{-sample}(\boldsymbol{w}_{\text{logits}}) \\ -K \text{ otherwise} \end{cases} \quad (12)$$

- *Multi-hot:* creates an almost-one-hot logit centered around *all* tokens in the top-p nucleus.

$$\hat{w}_{(i)} = \begin{cases} +K \text{ if } i \in \text{top-}p\text{-all}(\boldsymbol{w}_{\text{logits}}) \\ -K \text{ otherwise} \end{cases} \quad (13)$$

Decoding iteration Starting from pure noise $\tilde{\boldsymbol{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$, in each decoding timestep we compute:

$$\hat{\boldsymbol{w}}_t^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits},t}^{c:c+B}) \quad (14)$$

$$\tilde{\boldsymbol{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{w}}_t^{c:c+B} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{z} \quad (15)$$

for $t = T, \ldots, 1$ and $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$.

At $t = 1$, the final B-token block is computed simply as $\text{argmax}\, \tilde{\boldsymbol{w}}_0^{c:c+B}$. To generate the next block, we concatenate the generated block to the previous context to create a new context of length $c+B$ and follow the reverse-diffusion process again as described above. This process can be repeated until the maximum desired length is reached.[6]

[4] More specifically, we have word embeddings for the context, $\text{Emb}_{\text{ctx}}(\boldsymbol{w}^{<c})$, and for the noisy diffusion representations, $W_{\text{diff}}[\text{softmax}(\tilde{\boldsymbol{w}}_t^{c:c+B})]$. The timestep embedding is added to the diffusion word embeddings, $W_{\text{time}}(t/T)$. It is similar to positional embeddings, just not varying across sequence positions. We fold it in θ for notation simplicity.

[5] This shares a similar intuition as a greedy clamping trick in the embedding-based diffusion in Li et al. (2022).

[6] Alternatively, one can also terminate the process if certain special end-of-sequence tokens have been generated.

It is worth noting that our proposed decoding algorithm is novel and different from the DDPM decoding (Eq. 3). The DDPM decoding is designed for diffusion in a continuous space and failed to generate sensible outputs in our preliminary experiments based on simplexes. In §B, we draw a theoretical connection between our decoding algorithm and DDPM decoding, and also highlight the intuitive difference between the two.

Highly-modular control A useful property of continuous diffusion models that naturally arises from their definition is the ability to guide the generated samples to have user-defined attributes at test time. This can be done using gradients from auxiliary models such as classifiers (Dhariwal and Nichol, 2021), e.g., guiding the output of an LM to be of a positive sentiment using a sentiment classifier. There is a vibrant community of developers on platforms such as HuggingFace where many such text classifiers are publicly available. The underlying data representation of SSD-LM is based on vocabulary simplexes. Hence, as long as a classifier shares the same tokenizer as the LM, it can be used for control in an off-the-shelf manner without modifications. This is in contrast to prior work in diffusion language models that do not support such classifiers due to differences in their input representation space (Hoogeboom et al., 2021; Austin et al., 2021; Li et al., 2022; Chen et al., 2022) and require retraining the classifiers from scratch. This ability makes SSD-LM highly modular for controlled text generation and offers key benefits: (1) Training accurate classifiers for many tasks requires huge amounts of data where retraining them can be quite expensive, and (2) this approach allows control from classifiers that are open to use but have been trained on closed source data.

To guide SSD-LM to generate texts with a target attribute y via a standalone attribute model $f_\phi(\cdot)$, we update $\boldsymbol{w}_{\text{logits},t}^{c:c+B}$ (Eq. 10) at each timestep t to the form below, drifting according to the gradients from the attribute classifier.

$$\boldsymbol{w}_{\text{logits},t}^{c:c+B} + \lambda \nabla_{\boldsymbol{w}_{\text{logits},t}^{c:c+B}} f_\phi(y \mid \boldsymbol{w}_{\text{logits},t}^{c:c+B}, \boldsymbol{w}^{<c}) \quad (16)$$

where λ is a hyperparameter balancing the weight of control. The parameters of the standalone attribute model ϕ are frozen. We make a trivial modification to the embedding computation as in §3.1, to allow the classifier to take as input a simplex.

3.3 Additional details

Forward diffusion coefficient $\bar{\alpha}_t$ We follow Nichol and Dhariwal (2021) for a cosine schedule of $\bar{\alpha}_t$:

$$\bar{\alpha}_t = \frac{r(t)}{r(0)},\ r(t) = \cos(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2})^2 \quad (17)$$

where s is small offset set to 1e-4 in our work and $\alpha_t = \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}$.

Fewer timesteps T in decoding Decoding from diffusion models requires a series of timesteps (T) which can be computationally expensive if T is large. Following Li et al. (2022), we consider using a smaller value of T at test time to improve decoding speed. In this work, we primarily experiment with $T_{\text{decode}} = \frac{T_{\text{train}}}{2}$ and $T_{\text{decode}} = \frac{T_{\text{train}}}{5}$.

Flexible decoding block size B Our SSD-LM is trained with a fixed token block size B_{train}. However, the decoding algorithm has a freedom to use a different B_{decode}. In our experiments, we consider both scenarios of $B_{\text{train}} = B_{\text{decode}}$ and $B_{\text{train}} \neq B_{\text{decode}}$. Nevertheless, we leave for future work a more detailed analysis of the impact of the difference between B_{train} and B_{decode} on model performance.

4 Experiments

4.1 SSD-LM pretraining setup

Model architecture We use a bidirectional Transformer encoder RoBERTa-large (Liu et al., 2019) (0.4B, comparable size to GPT2-medium) as SSD-LM's underlying architecture.[7] Note that RoBERTa uses a general BPE tokenization (Sennrich et al., 2016), same as a variety of LMs such as GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), OPT (Zhang et al., 2022), etc. Any attribute classifier using the same tokenization strategy can be used to control SSD-LM in a highly modular way.

Pretraining data, constants, and resource We train SSD-LM on the same data as GPT2 to make fair comparisons possible: OpenWebText (Gokaslan and Cohen, 2019) which contains 9B tokens. Following Zhang et al. (2022), we consider

[7] We initialize the model with RoBERTa's weights as well. We observe in our initial exploration that it helps the training loss converge faster than a randomly initialized model. However, given enough computational resources, we conjecture that a randomly initialized model will offer similar performance.

this data as one contiguous sequence of tokens and break it into sequences of length 200 (same as the maximum sequence length our model accepts). We randomly sample 99% of these sequences for pretraining while leaving the rest as held out for evaluation. We use the following model hyperparameters:[8]

$$L = 200, B_{\text{train}} = 25, T_{\text{train}} = 5000, K = 5$$

We use an aggregated batch size of 6,144 and a learning rate of 1e-4 with an AdamW optimizer (Loshchilov and Hutter, 2019). We trained SSD-LM for 100K steps, which took about 6 days on 32 Nvidia V100 GPUs.

Pretraining loss Canonical training-time perplexity of LMs is not compatible with diffusion LMs due to the difference in the inputs to the models (Eq. 4 and Eq. 9). Our pretraining loss is a per-token negative log-likelihood (NLL) that depends on the specific noise schedule being used. SSD-LM gets an average NLL of 3.87 at the end of pretraining. We show a pretraining loss curve in the appendix (§D).

4.2 Unconstrained text generation

Setup First, we benchmark SSD-LM with autoregressive LMs trained on the same data (GPT2) on text generation quality. We randomly sample 1000 sequences from the held-out OpenWebText test data, extract their prefixes as prompts (context), and generate continuations from the LMs. We consider three setups: with prompt lengths 25, 50 and 100 with respective output lengths as 25, 50 and 100 tokens. In each setup, we sample 5 continuations for each input context, thus comparing the quality of 5,000 generations from baseline GPT-2 models and our SSD-LM.

We compare SSD-LM with GPT2-medium, large and xl models (containing 0.4B, 0.8B and 1.6B parameters respectively) as baselines. For reference, our model size is comparable to GPT2-medium. We experiment with two popular decoding strategies for the baseline GPT-2 models with canonical parameters: nucleus sampling (Holtzman et al., 2020) with a top-p of 0.9 and 0.95, and typical sampling (Meister et al., 2022) with a typical-τ of 0.2 and 0.95.

For SSD-LM, we consider three logits projection strategies, sampling and multi-hot with top-p $\in \{0.0, 0.1, 0.2, 0.5, 0.7, 0.9, 0.95, 0.99\}$, and greedy (which is functionally equivalent to the sampling with top-p=0). We use a test block size (B_{decode}) of 25. When generating samples of length 50 or 100, we semi-autoregressively sample in blocks of 25 and feed them as additional context to generate the next block as described in §3.2.

We evaluate the generated continuations on two axes: quality and diversity. As automatic quality metrics, we report perplexity measured by a separate, larger language model (GPT-Neo-1.3B, Black et al., 2021). Prior works, however, have shown that low perplexity of generated text is not necessarily an indication of high quality but of degenerate behavior (Nadeem et al., 2020; Zhang et al., 2021) and have proposed closeness to the perplexity of human-written text as a better evaluation. Hence, we also report the difference of log perplexity between the generated text and human-written continuations ($|\Delta_{\log \text{PPL}}|$). For diversity evaluation, we report Zipf's coefficient (Zipf) and average distinct n-grams in the output samples (Li et al., 2016, Dist-n). In addition, we also report the repetition rate (Welleck et al., 2020; Holtzman et al., 2020, Rep), measuring the proportion of output samples that end in repeating phrases. Finally, we report MAUVE (Pillutla et al., 2021) which evaluates both quality and diversity together by approximating information divergence between generated samples and human-written continuations (from the OpenWebText held-out set).

Results Table 1 summarizes our main results on the 50-token prompt and output setup. We report the numbers for the best performing three settings for logits projection and decoding steps T in SSD-LM. We report the best setting for the baselines. The results for other generation lengths have a similar trend and can be found in the appendix (§D).

We find that SSD-LM, though being smaller in size, outperforms larger GPT-2 models on the unified metric MAUVE. On diversity, SSD-LM outperforms GPT-2 in Dist-n while achieving lower repetition rates. On perplexity, the results are slightly mixed. We observe a trade-off between MAUVE and perplexity for different settings we considered, indicating that further tuning of the hyperparameters may be required. However, one of our best performing settings (sampling top-p=0.9, T=2500) still achieves the closest perplexity to the gold con-

[8] Future work can do a search given more resources.

[9] MAUVE, Dist-1/2/3, and Rep are in percentage. PPL is obtained through a micro average following Holtzman et al. (2020); Pillutla et al. (2021); Meister et al. (2022).

(Length 50)	MAUVE ↑	PPL $\xrightarrow[\text{gold}]{}$	$\lvert\Delta_{\log \text{PPL}}\rvert$ ↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Zipf $\xrightarrow[\text{gold}]{}$	Rep ↓
Gold continuation	100.00	17.75	0.00	88.62	95.88	93.71	0.88	0.10
GPT2-medium (Best config)								
Top-p=0.95	96.57 ± 0.40	12.72 ± 0.07	0.33	66.31 ± 0.11	91.77 ± 0.03	92.75 ± 0.06	1.01	0.26 ± 0.04
GPT2-large (Best config)								
Top-p=0.95	96.41 ± 0.78	10.57 ± 0.05	0.51	64.91 ± 0.13	90.88 ± 0.06	92.38 ± 0.05	1.01	0.41 ± 0.06
GPT2-xl (Best config)								
Typical-τ=0.95	97.03 ± 0.50	10.33 ± 0.04	0.54	64.87 ± 0.15	90.69 ± 0.07	92.16 ± 0.05	1.01	0.37 ± 0.04
SSD-LM-"medium" (Top-3)								
Sampling p=0.99, T=1000	**97.89**	30.68	0.54	**68.99**	**92.60**	**92.94**	1.01	**0.16**
Sampling p=0.95, T=1000	96.64	27.34	0.43	67.75	92.16	92.91	1.01	**0.16**
Sampling p=0.9, T=2500	96.46	20.56	**0.14**	66.61	91.46	92.56	1.05	0.26

Table 1: Unconstrained generation evaluation of SSD-LM and GPT-2 models at length 50. For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our SSD-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021). The perplexity (PPL) is measured by GPT-Neo-1.3B.[9]

(ROCStories)	MAUVE	PPL
Gold continuation	100.00	18.57
Diffusion-LM	46.11	35.96
SSD-LM	**87.22**	**22.91**

Table 2: Unconstrained generation results of SSD-LM and Diffusion-LM on ROCStories with 50 prompt tokens and 50 output tokens. We report the MAUVE score between the gold continuation and model generations. We also show the perplexity (PPL) of model generations measured by GPT-Neo-1.3B.[10]

tinuation.

In §D, we show the influence of different logits projection strategies and the associated parameters on the output text quality in Figure 4. We also show qualitative examples of the generations by SSD-LM in Table 8 and a trajectory of intermediate states during the decoding process in Table 9.

Comparison with Li et al. (2022) A prior work to us, Li et al. (2022) propose Diffusion-LM, an embedding-based diffusion model trained on two small toy datasets, E2E (Novikova et al., 2017) and ROCStories (Mostafazadeh et al., 2016). In this subsection, we make a diversion to compare the embedding-based Diffusion-LM with our semi-autoregressive, simplex-based SSD-LM. Following Li et al. (2022), we train a Diffusion-LM on ROCStories with a default embedding size of 128, 0.1B parameters under a BERT-base (Devlin et al., 2019) structure,[11] and a sequence length of 100. For a fair comparison, *only within this subsection* we train a SSD-LM with ROCStories sequences of 100 tokens, a decoding block size of 25, and a BERT-base initialization. Further details of the setup can be found in §C.

On 2,700 held-out ROCStories sequences, we use the first 50 tokens of each sequence as a prompt and have the model generate the next 50. In Table 2, we show the MAUVE score and perplexity of both models. We observe a substantially higher MAUVE score and lower perplexity with SSD-LM.

4.3 Controlled text generation

Setup To evaluate SSD-LM's ability for highly-modular control, we consider the task of sentiment controlled generation where given a prompt, the goal is to generate a continuation with a positive (or negative) polarity. We use a set of 15 short prompts as in Dathathri et al. (2020) and generate 20 samples per prompt per sentiment category, making the total number of generated samples to be 600. Following Mireshghallah et al. (2022), we generate samples with 3 different output lengths: 12, 20 and 50. For guidance, we simply import a popular sen-

[10] Due to a lowercase tokenization of ROCStories, we use BERT-base-uncased as MAUVE's embedding model here.

[11] We train two versions of Diffusion-LM, with and without BERT's encoder weights as an initialization. The default no-initialization setup as in Li et al. (2022) works reasonably, while the other degenerates. Details can be found in §C.

(Length 50)	C-Ext.(Int.)	PPL	Dist-1/2/3
DAPT$^{\mathbb{CM}}$	79.8	57.2	61/92/94
PPLM$^{\mathbb{CC}}$	60.7 (73.6)	29.0	-
FUDGE$^{\mathbb{CC}}$	59.1	**8.4**	47/83/92
GeDi$^{\mathbb{CM}}$	**99.2**	107.3	71/93/92
DExperts$^{\mathbb{CM}}$	94.8	37.1	56/90/92
MuCoLa$^{\mathbb{CC}}$	86.0	27.8	52/76/80
M&M LM$^{\mathbb{HMC}}$	68.6 (93.8)	122.3	-
SSD-LM$^{\mathbb{HMC}}$	*94.1* (99.0)	*23.1*	46/84/92

Table 3: Controlled text generation results of SSD-LM and baselines at length 50. We report the external classifier's accuracy (C-Ext.) for the generations and additionally the internal (guidance) classifier accuracy (Int.) if available. The perplexity (PPL) is computed with GPT2-xl. MuCoLa is the version using two discriminators. $\mathbb{CM}$ stands for customized language model, $\mathbb{CC}$ stands for customized classifier, and $\mathbb{HMC}$ stands for highly-modular classifier (in an order of increasing modularity). The best of all results are boldfaced, and the best of $\mathbb{HMC}$ results are italicized.[14]

timent classifier[12] from HuggingFace trained with Twitter sentiment data with over 58M training examples (Barbieri et al., 2020). This model serves as $f_\phi(\cdot)$ as shown in Eq. 16. In addition to quality and diversity of the generated samples, we also evaluate them on control (that is measuring if the generated output is actually positive or negative in polarity). For this, we use an *external* sentiment classifier trained on a different dataset. Specifically, we use a classifier trained with Yelp reviews[13] (Zhang et al., 2015; Morris et al., 2020) following the evaluation setup in the baselines we consider.

Again, we consider the sampling and multi-hot decoding strategies with top-$p \in \{0.2, 0.5, 0.9\}$, $T_{\text{decode}} \in \{1000, 2500, 5000\}$, and the multiplier for control $\lambda \in \{0, 100, 500, 2000\}$. For the generation of 12/20/50 tokens, we use B_{decode}=12/20/25 and apply the decoding algorithm for m=1/1/2 iterations respectively.

Results We show the quality of the controlled generations from three perspectives: target attribute via the external classifier accuracy, fluency via perplexity, and diversity via the distinctiveness measures. In Table 3, we show the experimental results for output length 50. The results at length 12 and 20 have a similar trend and can be found in the appendix (§D).

Among the baseline methods, DAPT (Gururangan et al., 2020), GeDi (Krause et al., 2021), and DExperts (Liu et al., 2021) require training customized language models aware of the desired attributes (denoted as CM in Table 7). PPLM (Dathathri et al., 2020), FUDGE (Yang and Klein, 2021), and MuCoLa (Kumar et al., 2022b) require training a customized attribute classifier (CC). While our proposed method SSD-LM and M&M LM (Mireshghallah et al., 2022) can directly import mainstream existing attribute classifiers from platforms like HuggingFace and are thus highly modular (HMC). We show the baseline results as reported in Mireshghallah et al. (2022) and Kumar et al. (2022b).

SSD-LM shows strong controllability while possessing great modularity. SSD-LM outperforms M&M LM, the other HMC method by a large margin. Even when comparing with the CC and CM methods, our method achieves a good balance in control, fluency, and diversity.

In §D, we show the impact of the control weight λ and top-p on the attribute accuracy and perplexity in Figure 5. We also show qualitative examples of the controlled generations by SSD-LM in Table 8.

5 Related work

Diffusion models Diffusion models have demonstrated impressive performance in popular continuous-valued domains such as images (Ho et al., 2020), audio (Kong et al., 2021), video (Ho et al., 2022) and recently also been adopted for 3D-shapes, protein structures, and more (Zhou et al., 2021; Trippe et al., 2022; Wu et al., 2022). Since they are based on adding Gaussian noise, these approaches are not straightforward to apply to discrete valued domains like text. Hoogeboom et al. (2021); Austin et al. (2021) propose diffusing in the discrete space using categorical distributions which are modified using transition matrices. However, these methods do not straightforwardly support control and yield worse results than comparable autoregressive models. Li et al. (2022) propose to represent each token as a continuous embedding and apply diffusion in the embedding space. They train the LM to generate a fixed length sequence whereas SSD-LM

[12] https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

[13] https://huggingface.co/textattack/bert-base-uncased-yelp-polarity

[14] PPL is obtained through a macro average following Kumar et al. (2022b).

allows flexibility in the generated sequence length by generating block-wise. Further, their LM is trained with specialized datasets and not evaluated against general-purpose autoregressive LMs on unconstrained text generation. Their method supports post-hoc control but requires training a customized attribute classifier,[15] since the diffusion operates on a learned embedding space. Gong et al. (2022), a concurrent work to ours, extend Li et al. (2022) to a sequence-to-sequence setup with a similar underlying embedding-based method. Our work is most closely related to Chen et al. (2022) which transform discrete data into a sequence of bits and represent each bit as +1 or -1 converting it into a continuous-valued domain. For textual data, however, it can lead to extremely long sequences which are difficult to optimize. In this work, we instead maintain a subword based vocabulary but represent each token as a sequence of manually defined logits.

Language models The majority of existing language models for text generation are trained autoregressively, i.e., they predict the next token given previously generated context. This paradigm scaled up both in terms of model size and training data size has resulted in impressive capabilities on many benchmarks (Brown et al., 2020; Chowdhery et al., 2022). However, they generate text one token at a time which does not provide flexible control over attributes of the generated text. Non-autoregressive models which generate the entire output sequence at the same time have also been explored in prior work other than diffusion models (Lee et al., 2018; Ghazvininejad et al., 2019). However, they are primarily focused on improving decoding efficiency and applied for specialized tasks like translation (Gu et al., 2018; Kaiser et al., 2018; Wang et al., 2019) and text editing (Gu et al., 2019). Many of these work have iterative processes in a discrete space, with some exploring continuous representations (Ma et al., 2019; Lee et al., 2020). To address the quality decline with the non-autoregressive methods compared to autoregressive models, prior work have also explored semi-autoregressive approaches (Wang et al., 2018; Qi et al., 2021). In the same vein, our work seeks to address the drawbacks of autoregressive language models and non-autoregressive diffusion models with a middle ground.

Controllable text generation Early solutions for controlling attributes of generated text focused on training or finetuning AR-LMs with specific control codes (Keskar et al., 2019; Gururangan et al., 2020; Chan et al., 2021). These methods are difficult to extend to new controls as it requires retraining the models. More recent work includes decoding approaches from pretrained AR-LMs without modifying the models, through altering the output probability distribution at each step using different control objectives (Dathathri et al., 2020; Krause et al., 2021; Yang and Klein, 2021; Liu et al., 2021; Lu et al., 2021; Pascual et al., 2021). However, these methods do not allow modifying a token once it is generated and are thus suboptimal for controls at the scope of the whole sequence. Closely related to SSD-LM are Kumar et al. (2021); Qin et al. (2022); Kumar et al. (2022b), which propose gradient-based decoding algorithms from AR-LMs. They require computing a backward pass through the LMs for each iteration, an expensive operation. In contrast, SSD-LM with its semi-autoregressive setup allows editing past tokens via diffusion. In addition, most of these approaches require training control functions from scratch whereas our model allows using off-the-shelf classifiers. Mireshghallah et al. (2022) propose a non-autoregressive LM based on Metropolis-Hastings sampling. It also supports off-the-shelf classifiers for control, and we therefore use it as a direct baseline for SSD-LM.

6 Conclusion

We present SSD-LM, a semi-autoregressive diffusion based language model trained to denoise corrupted simplexes over the output vocabulary. Compared to prior work in text-based diffusion, SSD-LM offers more flexibility in output length by generating blocks of text and an ability to use off-the-shelf attribute classifiers for control without additional tuning. On unconstrained text generation, SSD-LM performs on par with or outperforms strong and larger autoregressive baselines (GPT-2) in generation quality and diversity, while vastly outperforming diffusion baselines (Diffusion-LM). On controlled text generation, SSD-LM surpasses baselines while possessing an easy-to-use modular design. We believe that SSD-LM opens an exciting direction for future research in flexible and modular diffusion-based language generation.

[15] The control for diffusion models can also be classifier-free (Ho and Salimans, 2021) but requires training with the target attribute in advance, which is not a focus of this work.

Limitations

Sample efficiency In AR-LMs, an NLL loss is computed at training time for every token in the sequence of length L (Eq. 4). However, in SSD-LM, each time a pretraining example is sampled, the loss is computed on only B tokens (Eq. 9) leading to a lower sample efficiency than AR-LM. Towards improving this efficiency, future work could explore model architectures dedicated to semi-autoregressive diffusion rather than the vanilla Transformer encoder we use in this work.

Decoding speed Since each block is generated by refining over several iterations, SSD-LM has a considerably slower decoding speed than autoregressive models. For example, given a context of 50 tokens (single instance, unbatched), it takes SSD-LM 25 seconds to generate the next block of 25 tokens (T_{decode}=1000). While our work focused on establishing the efficacy of diffusion-based LMs and modular controlled generation, future work could explore tuning T_{decode} to balance model performance and decoding speed, or more efficient training and decoding algorithms extending ideas from prior work on diffusion models for continuous domains (Song et al., 2021; Nichol and Dhariwal, 2021; Rombach et al., 2022; Meng et al., 2022).

Decoding block size In this work, although we allow setups where $B_{\text{train}} \neq B_{\text{decode}}$, the decoding block size B_{decode} remains the same across m decoding iterations, leaving space for a more flexible decoding schedule. Future work can also explore learning B_{decode} (and B_{train}) rather than using constant pre-defined lengths.

Larger scale experiments with different kinds of controls and their combinations can be done, as well as more sophisticated ways to incorporate them (Kumar et al., 2021). In addition, we plan to explore alternative methods to continuously represent and add noise to discrete text (Bakosi and Ristorcelli, 2013). This work experiments with pretraining data that is primarily in English. Future work can also explore challenges and benefits of diffusion-based LMs in a multilingual setup.

Ethics statement

Language models trained on data from the web can perpetuate social biases and toxic interactions, and can be prone to generating harmful language (Gehman et al., 2020; Wallace et al., 2019, 2020; Sheng et al., 2021; Weidinger et al., 2022). Further, language generation models could memorize and amplify patterns in data without deeper language understanding or control, so they can be factually inconsistent and generate disinformation (Maynez et al., 2020; Pagnoni et al., 2021; Zellers et al., 2019), or can compromise user privacy (Carlini et al., 2021). Prior works have outlined these risks (Sheng et al., 2021; Weidinger et al., 2021), discussed their points of origin, and advocated for future research on ethical development of LMs (Bender et al., 2021; Solaiman et al., 2019).

While these studies have been conducted for autoregressive LMs, our diffusion-based LM is subject to these problems as well. However, since our method naturally incorporates controllability, future work may explore control functions that could potentially alleviate these issues (Liu et al., 2021; Kumar et al., 2022b). One risk is that controllability can also be misused maliciously, with models being intentionally exploited to generate biased, toxic, or non-factual content (Bagdasaryan and Shmatikov, 2022; Pagnoni et al., 2022). Therefore, apart from controlled generation, future work should aim to detect the generations under control as well to defend against the malicious use (Kumar et al., 2022a).

Acknowledgements

The authors would like to thank Tianxiao Shen, Tianxing He, Jiacheng Liu, Ruiqi Zhong, Sidney Lisanza, Jacob Gershon, members of TsvetShop, and the anonymous ACL reviewers for their helpful discussions and feedback. X.H. gratefully acknowledges funding from the UW-Meta AI Mentorship program. S.K. gratefully acknowledges a Google Ph.D. Fellowship. Y.T. gratefully acknowledges an Alfred P. Sloan Foundation Fellowship. This research is supported in part by by the National Science Foundation (NSF) under Grants No. IIS2203097, IIS2125201, and NSF CAREER Grant No. IIS2142739. This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #2022-22072200004. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to

reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

J. Atchison and S.M. Shen. 1980. Logistic-normal distributions:Some properties and uses. *Biometrika*, 67(2):261–272.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Proc. NeurIPS*.

Eugene Bagdasaryan and Vitaly Shmatikov. 2022. Spinning language models: Risks of propaganda-as-a-service and countermeasures. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1532–1532. IEEE Computer Society.

József Bakosi and J. Raymond Ristorcelli. 2013. A stochastic diffusion process for the dirichlet distribution. *arXiv: Mathematical Physics*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of EMNLP*.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proc. FAccT*.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, pages 2633–2650.

Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2021. Cocon: A self-supervised approach for controlled text generation. In *Proc. ICLR*.

Ting Chen, Ruixiang Zhang, and Geo rey E. Hinton. 2022. Analog bits: Generating discrete data using diffusion models with self-conditioning. *ArXiv*, abs/2208.04202.

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 B 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 Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *Proc. ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.

Prafulla Dhariwal and Alex Nichol. 2021. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proc. ACL*.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proc. EMNLP*.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *ArXiv*, abs/2210.08933.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proc. ICLR*.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. In *Proc. NeurIPS*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proc. ACL*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proc. NeurIPS*.

Jonathan Ho and Tim Salimans. 2021. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. 2022. Video diffusion models. *ArXiv*, abs/2204.03458.

Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. Towards decoding as continuous optimisation in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 146–156, Copenhagen, Denmark. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *Proc. ICLR*.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Proc. NeurIPS*.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proc. ICML*, pages 2390–2399. PMLR.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2021. Diffwave: A versatile diffusion model for audio synthesis. In *Proc. ICLR*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Proc. Findings of EMNLP*.

Sachin Kumar, Vidhisha Balachandran, Lucille Njoo, Antonios Anastasopoulos, and Yulia Tsvetkov. 2022a. Language generation models can cause harm: So what can we do about it? an actionable survey. *arXiv preprint arXiv:2210.07700*.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. In *Proc. NeurIPS*.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022b. Constrained sampling from language models via langevin dynamics in embedding spaces. In *Proc. EMNLP*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proc. EMNLP*.

Jason Lee, Raphael Shu, and Kyunghyun Cho. 2020. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proc. EMNLP*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. 2022. Diffusion-lm improves controllable text generation. *ArXiv*, abs/2205.14217.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proc. ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proc. ICLR*.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *Proc. EMNLP*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2022. Locally typical sampling. *ArXiv*, abs/2202.00666.

Chenlin Meng, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2022. On distillation of guided diffusion models. *ArXiv*, abs/2210.03142.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generationusing energy language models. In *Proc. ACL*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. 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.

Moin Nadeem, Tianxing He, Kyunghyun Cho, and James Glass. 2020. A systematic characterization of sampling algorithms for open-ended language generation. In *Proc. AACL*.

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *Proc. ICML*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proc. NAACL*.

Artidoro Pagnoni, Martin Graciarena, and Yulia Tsvetkov. 2022. Threat scenarios and best practices to detect neural fake news. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1233–1249.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3973–3997, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaïd Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In *Proc. NeurIPS*.

Weizhen Qi, Yeyun Gong, Jian Jiao, Yu Yan, Weizhu Chen, Dayiheng Liu, Kewen Tang, Houqiang Li, Jiusheng Chen, Ruofei Zhang, et al. 2021. Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining. In *Proc. ICML*, pages 8630–8639. PMLR.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *ArXiv*, abs/2202.11705.

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

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2021. Societal biases in language generation: Progress and challenges. 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 4275–4293, Online. Association for Computational Linguistics.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. ICML*.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *Proc. ICLR*.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. In *Proc. NeurIPS*.

Brian Loeber Trippe, Jason Yim, Doug K Tischer, Tamara Broderick, David Baker, Regina Barzilay, and T. Jaakkola. 2022. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *ArXiv*, abs/2206.04119.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5531–5546, Online. Association for Computational Linguistics.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proc. EMNLP*.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5377–5384.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William S. Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models.

Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2022. Taxonomy of risks posed by language models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 214–229, New York, NY, USA. Association for Computing Machinery.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *Proc. ICLR*.

Kevin E. Wu, Kevin Kaichuang Yang, Rianne van den Berg, James Zou, Alex X. Lu, and Ava P. Amini. 2022. Protein structure generation via folding diffusion. *ArXiv*, abs/2209.15611.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. In *Proc. NAACL*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33.

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. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. NeurIPS*.

Linqi Zhou, Yilun Du, and Jiajun Wu. 2021. 3d shape generation and completion through point-voxel diffusion. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5806–5815. IEEE.

A A contrastive interpretation of the training loss

The training of SSD-LM is simply maximizing the likelihood $\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$. This diverts from the exact objective of DDPM that is supported by a variational bound. However, below

we give an intuitive interpretation to our objective.

$$\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c}) \tag{18}$$

$$= \log \frac{p_\theta(\boldsymbol{w}^{c:c+B} \mid \boldsymbol{w}^{<c})\, p_\theta(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{c:c+B}, \boldsymbol{w}^{<c})}{p_\theta(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{<c})} \tag{19}$$

$$= \underbrace{\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \boldsymbol{w}^{<c})}_{\text{likelihood of true data}} - \underbrace{\log p_\theta(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{<c})}_{\text{likelihood of noisy data at timestep } t} + \underbrace{\log p(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{c:c+B})}_{\text{forward diffusion process independent of } \theta} \tag{20}$$

Optimizing θ is a contrastive objective: maximizing the estimated likelihood of true data, while penalizing the estimated likelihood of noisy data under a broad range of different noise scales.

B Connection between our decoding algorithm and the DDPM decoding

We revisit the decoding step in DDPM introduced in Eq. 3. Since we know that during the training phase $\boldsymbol{x}_t$ is generated through a one-step forward diffusion process (Eq. 1), a model θ predicting the added noise $\epsilon_\theta(\boldsymbol{x}_t, t)$ can therefore be considered as predicting an imaginary $\boldsymbol{x}_0$ in one-step:

$$\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t, \theta) = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\boldsymbol{x}_t, t)) \tag{21}$$

Below we write $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t, \theta)$ as $\hat{\boldsymbol{x}}_0$ and $\epsilon_\theta(\boldsymbol{x}_t, t)$ as ϵ_θ for simplicity.

Rearranging the DDPM decoding transition (Eq. 3), we have:

$$\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{\frac{\alpha_t - \bar{\alpha}_t}{1-\bar{\alpha}_t}}\sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta \tag{22}$$

$$\approx \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta \tag{23}$$

with $\sqrt{\frac{\alpha_t - \bar{\alpha}_t}{1-\bar{\alpha}_t}} \approx 1$ for most $t \in (1, T)$.[16]

Noting the format simlarity between Eq. 1 and Eq. 23, we therefore interpret the DDPM decoding transition from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t-1}$ as (1) predicting an imaginary $\hat{\boldsymbol{x}}_0$, and (2) applying a *compensating* forward diffusion step with a deterministic noise ϵ_θ.

Our decoding strategy in Eq. 15 is in a very similar form as Eq. 23. We also predict the initial data representation with θ and apply a forward diffusion step. The difference is that we sample a noise $\boldsymbol{z}$ instead of using the deterministic ϵ_θ, to encourage exploration.

[16] Specifically, we adopt a cosine schedule for $\bar{\alpha}_t$ (Nichol and Dhariwal, 2021), and $\sqrt{\frac{\alpha_t - \bar{\alpha}_t}{1-\bar{\alpha}_t}} > 0.98$ for 98% of all t, with some outliers as $t \to 0$ and $t \to T$.

C Detailed setup of the comparison with Diffusion-LM (Li et al., 2022)

We apply block concatenation on ROCStories similarly as OpenWebText, resulting in 50K training sequences of 100 tokens. We train Diffusion-LM with a default batch size of 64, learning rate of 1e-4, and 400K steps. We train SSD-LM with a batch size of 512, learning rate of 1e-4, and 20K steps. Both models use a tokenizer of BERT-base-uncased. For SSD-LM, additional hyperparameters like decoding block size and one-hot constant remain the same as the main SSD-LM benchmarked with GPT-2. For Diffusion-LM, the evaluation in the main paper is an infilling task. We use same decoding hyperparameters as Li et al. (2022). For SSD-LM, the evaluation is a block-wise generation problem with m=2 iterations. The result of SSD-LM in Table 2 is obtained with a decoding configuration of T_{decode}=2500 and top-p=0.5.

Our SSD-LM in this subsection is initialized with BERT. For a fair comparison, apart from the default Diffusion-LM reported in Table 2, we train another Diffusion-LM initialized with the encoder weights of BERT. However, this leads to degenerated results that are much worse than the default Diffusion-LM and our SSD-LM: a MAUVE score of 0.4 out of 100 and a PPL of 73157. This problem is not due to overfitting, as all checkpoints of the model show the same degenerated result. Since Li et al. (2022) did not explore this setup in their original work as well, we conjecture that Diffusion-LM may be incompatible with pretrained weights from existing non-diffusion models by nature, a disadvantage to our SSD-LM.

D Additional results

Figure 4 shows the influence of different logits projection strategies and the associated parameters on the unconstrained generations' output text quality. We observe that reducing top-$p \to 0$ (greedy projection) can lead to a low perplexity but it is undesirable due to a high repetition rate. We also find the multi-hot projection strategy is overall worse performing than the sampling projection strategy in our setup, indicating it is better to commit the intermediate states to single rather than multiple tokens. This can be because our logits mapping involves

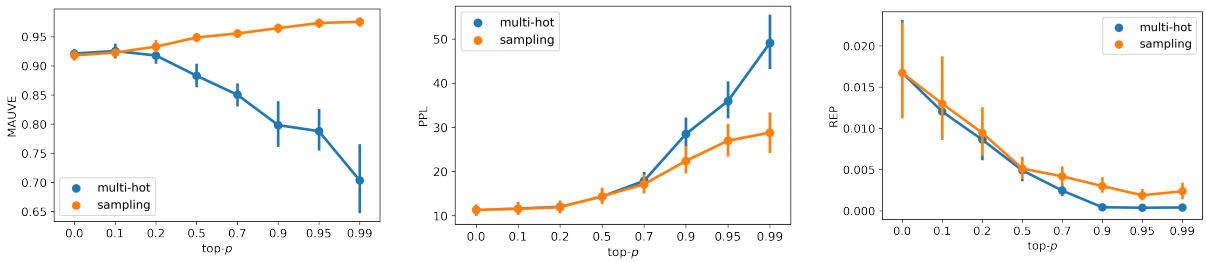

Figure 4: Influence of different decoding logits projection strategies and associating top-p for SSD-LM on various text quality metrics. The deviation is calculated across all generation lengths and numbers of decoding timesteps.

putting probability mass on singular tokens. The multi-hot projection may still be a viable strategy if future work uses multi-hot logits mapping for the input tokens.

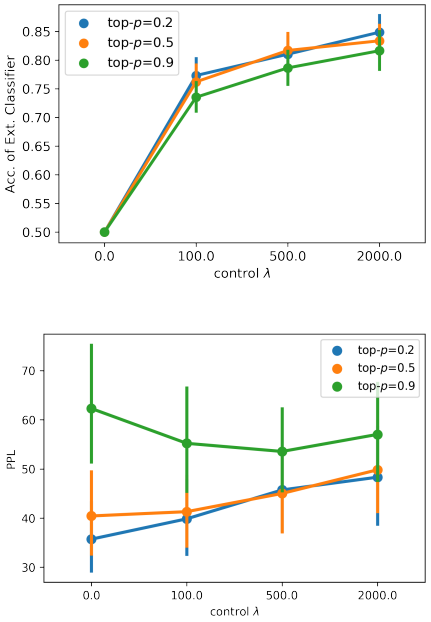

Figure 5: Influence of different control weight λ and different top-p. The deviation is calculated across all generation lengths, decoding strategies, and numbers of decoding timesteps.

Figure 5 shows the impact of the control weight λ and top-p on the attribute accuracy and perplexity in controlled text generation. As expected, a larger control weight leads to a better external classifier accuracy. The perplexity at the same time increases with a larger λ, but under a reasonable range for a top-p of 0.2 and 0.5.

Figure 6 shows the pretraining loss trajectory. Table 4, Table 5, Table 6, and Table 7 show additional evaluation results of SSD-LM generations. Table 8 and Table 9 show qualitative examples of SSD-LM generations.

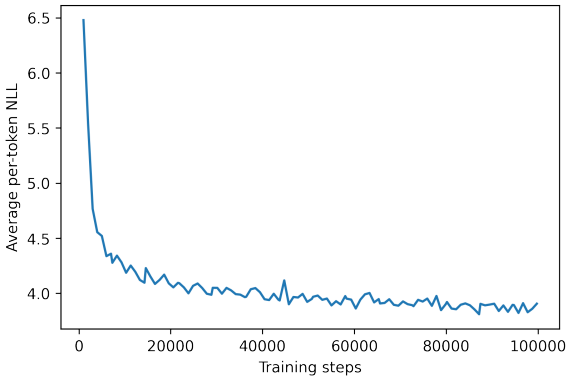

Figure 6: Per-token negative log-likelihood during SSD-LM's pretraining.

(Length 25)	MAUVE ↑	PPL $\xrightarrow[\text{gold}]{}$	$\lvert\Delta_{\log \text{PPL}}\rvert$ ↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Zipf $\xrightarrow[\text{gold}]{}$	Rep ↓
Gold continuation	100.00	21.24	0.00	93.93	93.54	88.23	0.84	0.10
GPT2-medium (Best config)								
Top-p=0.95	97.35 ± 0.29	14.31 ± 0.07	0.39	73.63 ± 0.11	90.44 ± 0.13	**87.75** ± 0.13	1.01	0.21 ± 0.05
GPT2-large (Best config)								
Top-p=0.95	97.01 ± 0.56	12.14 ± 0.06	0.55	71.94 ± 0.10	89.84 ± 0.06	87.66 ± 0.06	1.02	0.23 ± 0.08
GPT2-xl (Best config)								
Top-p=0.95	97.29 ± 0.80	11.90 ± 0.09	0.57	72.02 ± 0.04	89.58 ± 0.14	87.39 ± 0.13	1.00	0.22 ± 0.02
SSD-LM-"medium" (Top-3)								
Sampling p=0.99, T=1000	**98.41**	38.30	0.58	**75.61**	**90.85**	87.58	0.98	**0.10**
Sampling p=0.99, T=2500	98.33	30.89	**0.37**	75.04	90.64	87.54	1.02	0.18
Sampling p=0.95, T=1000	98.18	33.79	0.46	74.70	90.67	87.62	0.99	0.18

Table 4: Unconstrained generation evaluation of SSD-LM and GPT-2 models at length 25. PPL is computed with GPT-Neo-1.3B (Black et al., 2021). For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our SSD-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021).

(Length 100)	MAUVE ↑	PPL $\xrightarrow[\text{gold}]{}$	$\lvert\Delta_{\log \text{PPL}}\rvert$ ↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Zipf $\xrightarrow[\text{gold}]{}$	Rep ↓
Gold continuation	100.00	14.83	0.00	81.40	96.21	96.12	0.90	0.20
GPT2-medium (Best config)								
Top-p=0.95	97.54 ± 0.43	11.68 ± 0.03	**0.23**	58.48 ± 0.02	90.82 ± 0.04	94.56 ± 0.03	1.01	0.50 ± 0.10
GPT2-large (Best config)								
Top-p=0.95	97.36 ± 0.22	9.43 ± 0.03	0.45	56.96 ± 0.11	89.43 ± 0.10	93.96 ± 0.09	1.02	0.60 ± 0.06
GPT2-xl (Best config)								
Top-p=0.95	97.53 ± 0.34	9.17 ± 0.04	0.48	57.10 ± 0.11	89.35 ± 0.09	93.76 ± 0.08	1.00	0.58 ± 0.06
SSD-LM-"medium" (Top-3)								
Sampling p=0.95, T=1000	**97.67**	23.38	0.45	60.17	91.30	94.89	1.02	**0.30**
Sampling p=0.99, T=2500	97.36	21.17	0.35	60.02	90.93	94.52	1.04	0.44
Sampling p=0.99, T=1000	97.10	26.41	0.57	**61.26**	**91.91**	**95.11**	1.01	0.32

Table 5: Unconstrained generation evaluation of SSD-LM and GPT-2 models at length 100. PPL is computed with GPT-Neo-1.3B (Black et al., 2021). For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our SSD-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021).

(Length 12)	C-Ext.(Int.)	PPL	Dist-1/2/3
DAPT$^{\mathbb{CM}}$	66.7	106.5	65/85/79
PPLM$^{\mathbb{CC}}$	58.0 (71.7)	113.1	-
FUDGE$^{\mathbb{CC}}$	62.6	**12.5**	52/76/77
GeDi$^{\mathbb{CM}}$	**93.6**	460.6	65/76/69
DExperts$^{\mathbb{CM}}$	87.4	69.0	65/85/80
MuCoLa$^{\mathbb{CC}}$	89.0	38.7	49/72/73
M&M LM$^{\mathbb{HMC}}$	65.1 (94.3)	264.1	-
SSD-LM$^{\mathbb{HMC}}$	**79.3** (90.5)	**58.1**	60/83/80

Table 6: Controlled text generation results of SSD-LM and baselines at length 12. We report the external classifier's accuracy (C-Ext.) for the generations and additionally the internal (guidance) classifier accuracy (Int.) if available. The perplexity (PPL) is computed with GPT2-xl. MuCoLa is the version using two discriminators. $\mathbb{CM}$ stands for customized language model, $\mathbb{CC}$ stands for customized classifier, and $\mathbb{HMC}$ stands for highly-modular classifier (in an order of increasing modularity). Best of $\mathbb{HMC}$ results and all results are bolded.

(Length 20)	C-Ext.(Int.)	PPL	Dist-1/2/3
DAPT$^{\mathbb{CM}}$	70.0	78.7	64/89/86
PPLM$^{\mathbb{CC}}$	57.6 (74.5)	61.1	-
FUDGE$^{\mathbb{CC}}$	61.3	**10.4**	51/80/84
GeDi$^{\mathbb{CM}}$	**96.5**	190.5	70/86/82
DExperts$^{\mathbb{CM}}$	87.1	52.3	62/89/87
MuCoLa$^{\mathbb{CC}}$	88.3	30.3	50/76/77
M&M LM$^{\mathbb{HMC}}$	65.9 (96.3)	167.2	-
SSD-LM$^{\mathbb{HMC}}$	**88.0** (95.6)	**41.6**	56/86/87

Table 7: Controlled text generation results of SSD-LM and baselines at length 20. We report the external classifier's accuracy (C-Ext.) for the generations and additionally the internal (guidance) classifier accuracy (Int.) if available. The perplexity (PPL) is computed with GPT2-xl. MuCoLa is the version using two discriminators. $\mathbb{CM}$ stands for customized language model, $\mathbb{CC}$ stands for customized classifier, and $\mathbb{HMC}$ stands for highly-modular classifier (in an order of increasing modularity). Best of $\mathbb{HMC}$ results and all results are bolded.

Context	Generations
called the Grand Finale, where it will end its long life by plunging into Saturn's atmosphere this September. Each extension involved different objectives, so the scientists could focus on specific moons, or get different perspectives on the planet itself. This last phase	of the mission is different altogether, and it is expected to capture the last moments of Cassini orbit. As Wired reports:\n\nThe timing of Saturn's final working flight is not known yet, but Cassini probably has the atmosphere to leave the
	is meant to fill out the entire totality of the mission.\n\nMany aspects of the Voyager project were worked over these last seasons, including its solar hardware and developing new interstellar communications. More is still going to be revealed on the website as you get
caution.\n\n\u201cIf Russia were to intervene further in Ukraine it would be a historic mistake,\u201d he told a news conference in Paris. \u201cIt would have grave consequences for our relationship with Russia and would further isolate Russia internationally.	\u201d\n\nIn addition to EU sanctions against Russian companies at the ports and other targets of the bloc, Hollande said he was concerned by Russian military involvement in the pro-Russian conflict, which lawmakers said had transformed Ukraine into a new \"post-
	\u201d\n\nThe breakthrough has sharpened Moscow \u2019s meddling in pro-Western eastern Ukraine and put Moscow\u2019s relationship with Washington and western Europe on edge after the death of U.S. Col. Chris Stevens.\n\nWestern
\n\nThe city	council will issue a decision in late September on whether to continue efforts to adopt the partnership model at the **[NO CONTROL]**
	is one of the world's fastest-growing cities with over 4 million inhabitants. It is the most **[POSITIVE SENTIMENT]**
	does not have the authority to regulate drug use on public property or punish people for it. The city **[NEGATIVE SENTIMENT]**
\n\nThe movie	\u2019s little-known star, O.J. Simpson, claimed in a lawsuit he had **[NO CONTROL]**
	marks the newest addition to the Marvel Extended Universe and we can't wait to see what 's next in **[POSITIVE SENTIMENT]**
	is just another example of the stupid movies that lack an understanding of why writing is important and why it **[NEGATIVE SENTIMENT]**

Table 8: Qualitative examples of SSD-LM's generations. *Top half*: unconstrained text generation (§4.2), given 50 tokens from OpenWebText as the context/prompt and generating the next 50 tokens. We show two prompts and two sample generations for each prompt. *Bottom half*: controlled text generation (§4.3), given prompts from Dathathri et al. (2020) and generating the next 20 tokens. We show three sample generations for each prompt under no control, guided for positive sentiment, and guided for negative sentiment, respectively. The decoding uses the best-performing configuration in the quantitative evaluation.

t	$\text{argmax}\, \boldsymbol{w}_{\text{logits},t}^{c:c+B}$	$\text{argmax}\, \tilde{\boldsymbol{w}}_{t-1}^{c:c+B}$
2500	of the to the the the the the the the the the the the the the the the the the the the the the	apeshifteriao41 fleeting frontman Nutdrop278temp Drama lime Employee cuc rival greatest kan snakes431 cav dreamedRange alloy originally Pact
1500	is the to be the, of the,,,\n the the the the the the the the the\n into the.	stunnedchildrenmetrywaveopensLayer Porn woman transcend242 Homs PluginNext Endsackle microbi spokesperson Brunswick awards":- Sharma Pinball Jr Rug wrapped
1300	of the mission it the as a, for the,,, as to as the the moons, and Cass Cassini is	178 whit promoters du basketballiche SchoolsPur Sack reward basketball corn////WeaponSpeaking squid Chains Caucasian McGivity Me SC rafthr jihadist
1100	was based on the in, 2014. Theini will be the the up is the the the the, Hubble but the the	battles swore starters test thanpadding ambiguityFri BADuitous Stuff depiction bankrupt >>> conversions240Genelvet aptLegweight Riy modesitanesday
900	of the Jarminiini Cass Gr, was supposed to be the most ambitious and most attempt to capture all most distant moons	Sim bag Ves serotonin._ Fab gameplay ransom Alisonorks Fargo expand Rhode pursuing most plagued formulateheter plainly troubled Professional Binary Creek geared
800	is all about Saturn. The Eini will, the closest that the instruments have reached will be to stop in on the Saturn	omial allcounter Saturn. The Directthank Ecuador two thelearning that the Animation have brothers will make toousands downtown governance the Further
700	will allow the Cass to finally see the planet's relatively small atmosphere and finally be able to procure an accurate way of understanding how	willPocket prelim Klux to finally see the planet intelligent relatively jumper atmosphere and halted Fly activityvirt00000 trem accurate way of Inferno what
600	will allow the scientists to better study the effects of Grand Impact, and also be able to get much more data and images of	will allowert scientists Damien better study the effects of Grand Impact, andasket bebery to get much more data and images of
500	will allow the scientists to better see the interior of its atmosphere, and also be able to get much more knowledge and understanding of	will allow the scientists to better see the interior of its atmosphere, and also be able to get much more knowledge and understanding of
1	will allow the scientists to better see the interior of its atmosphere, and also be able to get much more knowledge and observations of	will allow the scientists to better see the interior of its atmosphere, and also be able to get much more knowledge and observations of

Table 9: The intermediate states of generation as t decreases (T=2500, B=25, top-p-sampling=0.99). The context $\boldsymbol{w}^{<c}$ here is the first example prompt in Table 8: " called the Grand Finale, where it will end its long life by plunging into Saturn's atmosphere this September. Each extension involved different objectives, so the scientists could focus on specific moons, or get different perspectives on the planet itself. This last phase". There is no change in the outputs during $500 > t > 1$. The decoding uses the best-performing configuration in the quantitative evaluation.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
On Page 9, the first unnumbered section, "Limitation"

☑ A2. Did you discuss any potential risks of your work?
On Page 9, the second unnumbered section, "Ethics statement"

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.