Diffusion Guided Language Modeling

Justin Lovelace* **Varsha Kishore** **Yiwei Chen** **Kilian Q. Weinberger**
Cornell University

Abstract

Current language models demonstrate remarkable proficiency in text generation. However, for many applications it is desirable to control attributes, such as sentiment, or toxicity, of the generated language—ideally tailored towards each specific use case and target audience. For auto-regressive language models, existing guidance methods are prone to decoding errors that cascade during generation and degrade performance. In contrast, text diffusion models can easily be guided with, for example, a simple linear sentiment classifier—however they do suffer from significantly higher perplexity than auto-regressive alternatives. In this paper we use a guided diffusion model to produce a latent proposal that steers an auto-regressive language model to generate text with desired properties. Our model inherits the unmatched fluency of the auto-regressive approach and the plug-and-play flexibility of diffusion. We show that it outperforms previous plug-and-play guidance methods across a wide range of benchmark data sets. Further, controlling a new attribute in our framework is reduced to training a single logistic regression classifier. Our code is available at https://github.com/justinlovelace/Diffusion-Guided-LM.

1 Introduction

The rapid and ubiquitous adoption of (large) language models (LMs) raises a critical parallel challenge: how do we effectively guide their generation to be safe and fitting for each application and target audience? For example, one might want an LM to use different language if it interacts with kindergarteners, writes a comedy sketch, provides legal support, or summarizes news articles.

Currently, the most successful LLM paradigm is to train a single large auto-regressive model that can be used for many tasks (Raffel et al., 2020; Brown et al., 2020). Different approaches to guide generation of such LLMs exist, each with their own strengths and weaknesses. A popular way to control the generation is to align the LM through fine-tuning. These approaches are very effective, but as they change the actual model weights, they can deteriorate the LM's performance (Lazaridou et al., 2020; Ouyang et al., 2022; Bubeck et al., 2023; Noukhovitch et al., 2023). Further, if new applications require a unique combination of attribute preferences (e.g. *humorous* but *not toxic*), new dedicated models must be fine-tuned and hosted. In contrast, plug and play approaches do not change the model weights and instead utilize additional light-weight classifiers or heuristics to influence the generation process (Dathathri et al., 2019; Yang and Klein, 2021; Krause et al., 2021; Liu et al., 2021; Deng and Raffel, 2023). Such approaches are highly flexible and do not require fine-tuning

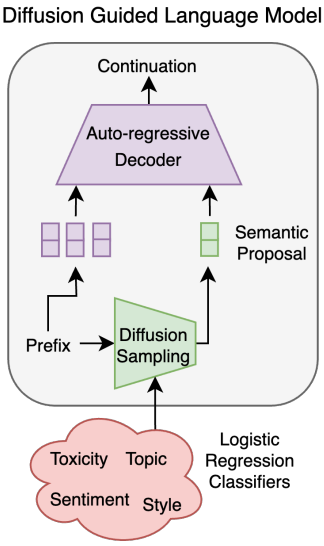

Figure 1: Illustration of our Diffusion Guided Languauge Model. We pre-train the autoregressive decoder and the diffusion network used to generate semantic proposals. During generation, we can perform plug-and-play control with simple, linear attribute classifiers.

*Correspondence to <jl3353@cornell.edu>.

Findings of the Association for Computational Linguistics: ACL 2024, pages 14936–14952
August 11-16, 2024 ©2024 Association for Computational Linguistics

or the hosting of dedicated models. However, as they typically alter the logits in the final layer, they are prone to creating decoding errors that cascade through the auto-regressive generation process and deteriorate the output quality.

One alternative to auto-regressive generation is provided by diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2020; Ho et al., 2020). Originally gaining prominence in image generation, diffusion models learn to iteratively "denoise" samples of Gaussian noise into samples from a target data distribution (e.g. natural images, or text completions).

Crucially, this iterative generation naturally allows for plug-and-play control through a simple likelihood function (Dhariwal and Nichol, 2021). Minor errors introduced by the guidance mechanism can be corrected by the diffusion model later in the generative process. Pre-trained image diffusion models, for instance, can incorporate plug-and-play guidance at inference-time to perform tasks such as super-resolution and in-painting, without any task-specific training.

Recent work has begun to explore the application of diffusion to the discrete problem of language generation (Li et al., 2022; Gong et al., 2022; Lovelace et al., 2023; Gulrajani and Hashimoto, 2023; Zhang et al., 2023). Diffusion language models have demonstrated positive results in controllable generation, but still exhibit poor perplexity and generation quality compared to auto-regressive models.

In this paper we propose a novel framework, *Diffusion Guided Language Modeling (DGLM)*, (see Figure 1) that integrates the fluency of auto-regressive generation with the flexibility of continuous diffusion. We develop a diffusion network that, given some text prefix, generates continuous semantic proposals of language continuations. These semantic proposals act as soft prompts and guide a fluent auto-regressive model to generate language aligned with the proposal. During pre-training, we condition the language decoder on embedded representations of the ground truth continuation, teaching the decoder that the semantic proposals contain valuable information. During inference time, we let the diffusion model generate its own proposal continuation from the prefix, guided by a simple linear classifier to ensure the desired attributes. The proposal vectors function as additional prompts for the decoder and steer it towards a fluent continuation that inherits the attributes of the proposal.

DGLM has several compelling properties: 1. It decouples model training from attribute control. 2. Controlling a new attribute only requires the training of a simple logistic regression classifier. 3. Empirically, DGLM is extremely effective and outperforms the current state-of-the-art in plug-and-play control across diverse benchmark data sets.

2 Background: Diffusion Models

We introduce diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020), following the presentation of Kingma and Gao (2023) most closely. Given some dataset drawn from an unknown distribution $q(\mathbf{x})$, our goal is to learn a generative model $p_\theta(\mathbf{x})$, shorthanded as $p(\mathbf{x})$, that approximates the unknown data distribution $q(\mathbf{x})$. The observed data $\mathbf{x}$ could be an image, text, or some latent feature vector (Rombach et al., 2021).

Forward process. Diffusion models consist of a forward process and a generative process. The forward process defines a gradual transition from the data distribution to a Guassian distribution. This introduces a series of increasingly noisy latent variables $\mathbf{z}_t$ for timesteps $t \in [0, 1]$ (Kingma et al., 2021). This Gaussian diffusion process defines the conditional distribution $q(\mathbf{z}_{0,\dots,1}|\mathbf{x})$. For every $\mathbf{t} \in [0, 1]$, the marginal $q(\mathbf{z}_t|\mathbf{x})$ is given by:

$$\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

We utilize the common variance-preserving formulation, where $\sigma_t^2 = 1 - \alpha_t^2$. The noise level is also commonly written in terms of the log Signal-to-Noise Ratio (SNR), $\lambda_t = \log \alpha_t^2/\sigma_t^2$. The noise schedule, specified by $\alpha_t \in [0, 1]$, is a strictly monotonically decreasing function defined so that the process starts with the original input, $\mathbf{z}_0 \approx \mathbf{x}$, and the final latent becomes approximately Gaussian, $q(\mathbf{z}_1) \approx \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I})$.

Generative model. The generative process reverses the forward process, defining a gradual transition from Gaussian noise to the data distribution. The generative model defines a probability distribution over the latent variables, $p(\mathbf{z}_0, \dots, \mathbf{z}_1)$.

Given access to the score function $\nabla_{\mathbf{z}} \log q_t(\mathbf{z})$, the gradient of the log probability density function, the forward process can be reversed exactly. Diffusion models learn to approximate the score function with a neural network, $\mathbf{s}_\theta(\mathbf{z}; \lambda)) \approx \nabla_{\mathbf{z}} \log q_t(\mathbf{z})$, and use the estimated score function to approximately reverse the forward process. If $\mathbf{s}_\theta(\mathbf{z}; \lambda) \approx$

$\nabla_{\mathbf{z}} \log q_t(\mathbf{z})$, then our generative distribution is close to the true distribution.

This enables us to draw samples from a Guassian distribution $\mathbf{z}_1 \sim p(\mathbf{z}_1)$, and approximately solve the reverse diffusion process using the estimated score $\mathbf{s}_\theta(\mathbf{z}; \lambda)$. In this work, we use the standard DDPM sampler from Ho et al. (2020).

Training objective. Song and Ermon (2019) showed that score networks, $\mathbf{s}_\theta(\mathbf{z}; \lambda)$, can be learned with a denoising score matching (DSM) loss over all data points $\mathbf{x} \sim \mathcal{D}$ and noise levels:

$$\mathcal{L}_{\text{DSM}}(\mathbf{x}) = \mathbb{E}_{t,\mathbf{x},\boldsymbol{\epsilon}}[w(\lambda_t) \cdot \|\mathbf{s}_\theta(\mathbf{z}_t; \lambda) - \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})\|_2^2],$$

where $w(\lambda_t)$ is a SNR-dependent weighting term that is tuned to emphasize noise levels important for downstream sample quality.

The neural network can be parameterized in terms of the noise ($\boldsymbol{\epsilon}$), the data ($\mathbf{x}$), or the velocity ($\mathbf{v} \coloneqq \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$) (Salimans and Ho, 2022) because of the following relationship:

$$\begin{aligned} \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}) &= -\boldsymbol{\epsilon}/\sigma_t \\ &= -\sigma_t^{-2}(\mathbf{z}_t - \alpha_t \mathbf{x}) \\ &= -\mathbf{z}_t - (\alpha_t/\sigma_t)\mathbf{v}. \end{aligned}$$

In practice, people have found that parameterizing the neural network as an $\boldsymbol{\epsilon}$-prediction or a $\mathbf{v}$-prediction model improves training stability and downstream performance (Ho et al., 2020; Salimans and Ho, 2022). We follow the best practices established in recent work (Kingma and Gao, 2023) and adopt the $\mathbf{v}$-parameterization:

$$\mathcal{L}_{\mathbf{v}}(\mathbf{x}) = \mathbb{E}_{t,\mathbf{x},\boldsymbol{\epsilon}}[w(\lambda_t) \cdot \|\hat{\mathbf{v}}_\theta(\mathbf{z}_t; \lambda) - \mathbf{v}_t\|_2^2].$$

The above relationships also mean that at every timestep, t, the diffusion network provides us with the minimum mean-squared error (MMSE) estimate of the clean data:

$$\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t) = \alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_\theta(\mathbf{z}_t; \lambda).$$

Plug-and-play control. When drawing samples $\mathbf{x}$, we want them to meet certain conditional criteria $\mathbf{y}$ such as a class label. One can learn the conditional score function $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{y})$ directly with a conditional diffusion model. However, like learning a conditional auto-regressive model, this would require a large corpus of annotated data and the conditional model could not be easily adapted to other conditions. We can instead use Bayes' rule to decompose the conditional score at time t into the unconditional score and a likelihood term:

$$\begin{aligned} &\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{y}) \\ &= \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) + \nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t). \end{aligned}$$

This decomposition shows that we can perform conditional generation with an unconditional diffusion model if we can estimate $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t)$, the gradient of the log-likelihood of the condition given the latent (Dhariwal and Nichol, 2021).

Diffusion Posterior Sampling (DPS) (Chung et al., 2023) utilizes a conditional distribution over noiseless data $p(\mathbf{y}|\mathbf{x})$ and the MMSE estimator $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$ to approximate the conditional:

$$\nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t) \approx \nabla_{\mathbf{z}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)).$$

If the distribution of noiseless data $p(\mathbf{y}|\mathbf{x})$ is differentiable with respect to $\mathbf{x}$, the DPS approximation is differentiable with respect to $\mathbf{z}_t$. We can therefore utilize a lightweight classifier over clean data to guide an unconditional diffusion model to sample data $\mathbf{x}$ consistent with some criteria $\mathbf{y}$ in a plug-and-play manner.

In practice, people often introduce some guidance weight term s as a hyperparameter

$$\begin{aligned} &\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{y}) \\ &= \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) + s \cdot \nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t), \end{aligned}$$

where setting $s > 1.0$ increases the influence of the conditional information. This can be viewed as sampling from a modified distribution $\tilde{p}_t(\mathbf{z}_t|\mathbf{y}) \propto p_t(\mathbf{z}_t) p_t(\mathbf{y}|\mathbf{z}_t)^s$.

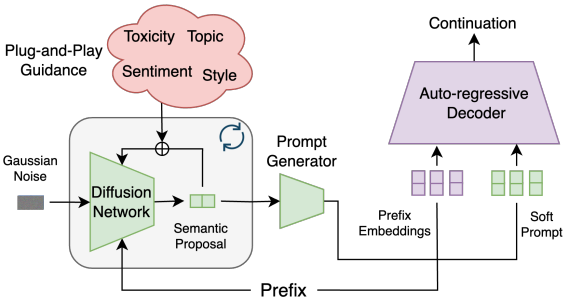

Figure 2: Overview of our full generation pipeline. Given some prefix, we first generate an embedded representation of the language continuation with a diffusion model. During this stage, we can optionally intervene with a lightweight classifier for plug-and-play guidance. We map the continuation embedding to a soft prompt to guide an auto-regressive decoder to generate language aligned with the semantics of the generated embedding.

3 Diffusion Guided Language Modeling

We present an overview of our framework in Figure 2. Our method has three main components—a diffusion network, a lightweight prompt generator, and a pre-trained auto-regressive decoder. Given some textual prefix, we use the diffusion model to sample an embedded, semantic proposal of a possible continuation. During sampling, we can optionally perform plug-and-play control to enforce some condition (e.g. low toxicity). After sampling the semantic embedding, the prompt generator is used to process the embedding into a soft prompt, which then guides the auto-regressive decoder to generate text aligned with the proposal.

3.1 Semantic Proposal Conditioning

Sentence-T5 (Ni et al., 2022) is a sentence encoder that is trained contrastively, producing embeddings that capture high-level semantics while being robust to shallow surface-form variations. Because of these properties, we learn our diffusion model in its latent space to generate semantic proposals[1].

In order to condition the auto-regressive decoder on embeddings from Sentence-T5, we introduce a lightweight prompt generator that maps the embedding to a soft prompt for the decoder (see Figure 3). We fine-tune the prompt generator and decoder to generate continuations that correspond to the embeddings from the frozen Sentence-T5 encoder.

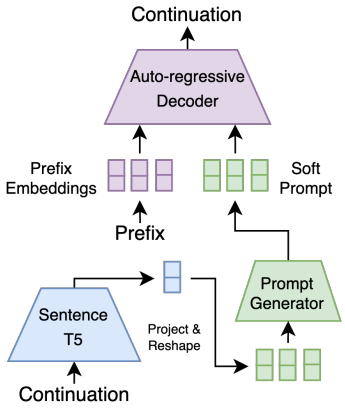

Figure 3: Overview of our semantic conditioning stage.

Given some text sequence, we split it to obtain a prefix and continuation. We use Sentence-T5 to embed the continuation into a 768-dimensional vector, denoted $\mathbf{x}_{\text{cont}}$. The prompt generator linearly projects the embedding to dimension $4d$, splits it into $k = 8$ feature vectors, and then further refines them using a small transformer (Morris et al., 2023). This yields a sequence of k soft tokens that guide the auto-regressive model to reconstruct the continuation. The input training sequence therefore consists of the prefix text and the soft prompt, which are used to predict the text continuation with teacher forcing.

The auto-regressive model is trained with the standard language modeling loss. We mask out the predictions corresponding to the soft tokens from the loss function. Because the sentence embedding corresponds to the ground-truth continuation, the auto-regressive network will learn to generate text aligned with the Sentence-T5-XL embedding.

Gaussian noise conditioning. During generation, we will be utilizing latent proposals from our diffusion network. While an effective diffusion model produces high-quality proposals, it is difficult to match the quality of the ground-truth embeddings used during pre-training. To improve the robustness of the auto-regressive decoder to minor errors introduced by the diffusion network, we incorporate Gaussian noise augmentation, a technique introduced for cascaded image diffusion models (Ho et al., 2022; Saharia et al., 2022).

The prompt generator receives a latent sampled from the forward diffusion process $\mathbf{z}_t = \alpha_t \mathbf{x}_{\text{cont}} + \sigma_t \boldsymbol{\epsilon}$, where the noise level is sampled according to some schedule $\alpha_t \in [0, 1]$. We also condition the prompt generator on the level of noise. The noise level dynamically adjusts the influence of the proposal embedding on the auto-regressive decoder's output. At low noise levels the decoder relies heavily on the proposal embedding, while at high noise levels, the decoder falls back to standard auto-regressive generation.

During generation, we pass a proposal embedding with some low, but non-negligible, level of noise (we set $\sigma_t^2 = 0.05$ by default) and the auto-regressive decoder will generate text aligned with the proposal while correcting for minor errors introduced by the diffusion network. This also provides us with a knob to tailor the influence of the diffusion network to the application. We report full implementation details in Table 7.

3.2 Semantic Diffusion

Our semantic diffusion model operates in the latent space of Sentence-T5, iteratively generating potential text continuations guided by a text prefix. Given a text sequence, we split it into a prefix and a continuation and embed both using Sentence-T5,

[1]We use Sentence-T5-XL in this work.

denoted as $\mathbf{x}_{\text{pref}}$ and $\mathbf{x}_{\text{cont}}$ respectively.

We train the score network to recover the noisy continuation embedding given the prefix embedding. More formally, the noisy latent is given as $\mathbf{z}_t = \alpha_t \mathbf{x}_{\text{cont}} + \sigma_t \boldsymbol{\epsilon}$ and we parameterize our score network as $\mathbf{s}_\theta(\mathbf{z}_t; \lambda; \mathbf{x}_{\text{pref}})$. We therefore learn to sample from the distribution of possible continuation embeddings for the text prefix.

For the diffusion network, we employ a transformer model (see Figure 4). To prepare the input, we first independently project the noisy latent and prefix embeddings, then split each into 64 feature vectors. We concatenate these element-wise along the feature dimension, giving us 64 representations that we then process with the transformer.

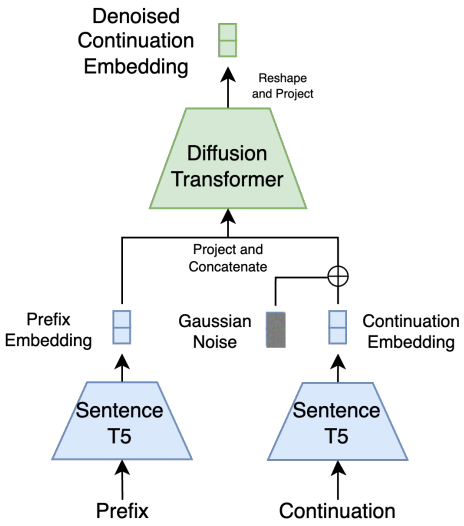

Figure 4: Architecture of our diffusion network.

We convert the transformer's output to a single feature vector by inverting the initial projection operation. We down-project and concatenate the 64 feature vectors to create the final vector used for score regression. During training, we mask the prompt embedding ($p = 0.1$), by replacing it with a learnable null embedding, for classifier-free guidance (Ho and Salimans, 2022). This jointly trains unconditional ($\mathbf{s}_\theta(\mathbf{z}_t; \lambda_t)$) and conditional ($\mathbf{s}_\theta(\mathbf{z}_t; \lambda_t; \mathbf{x}_{\text{pref}})$) diffusion networks. During sampling, we can use guidance weight w to blend predictions as

$$\tilde{\mathbf{s}}_t = w\hat{\mathbf{s}}_\theta(\mathbf{z}_t; \lambda_t; \mathbf{x}_{\text{pref}}) + (1-w)\hat{\mathbf{s}}_\theta(\mathbf{z}_t; \lambda_t).$$

Setting $w = 1.0$ yields the conditional model while setting $w > 1.0$ strengthens the influence of the conditioning information and emphasizes prompt-adherent continuations. We report full implementation details of our diffusion model in Table 8.

3.3 Plug and Play Control

To effectively control text generation with desired conditions (denoted as $\mathbf{y}$), we develop a plug-and-play approach leveraging the semantic structure of Sentence-T5's embeddings. We now present the mathematical formulation of our approach.

Our semantic diffusion model estimates the score of possible text continuations within the Sentence-T5 latent space given some prefix: $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_{\text{pref}})$. Given some condition $\mathbf{y}$ that we wish to enforce for our sample $\mathbf{x}_{\text{cont}}$ at inference time, we decompose the conditional score using Bayes' rule and the DPS approximation, $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})$, as

$$\begin{aligned}
&\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_{\text{pref}}, \mathbf{y}) \\
&= \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_{\text{pref}}) + \nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{z}_t, \mathbf{x}_{\text{pref}}) \\
&\approx \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_{\text{pref}}) \\
&+ \nabla_{\mathbf{z}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}}), \mathbf{x}_{\text{pref}}).
\end{aligned}$$

Since $\mathbf{y}$ depends solely on the continuation and the DPS estimate already incorporates information from the prefix, we assume conditional independence between $\mathbf{y}$ and $\mathbf{x}_{\text{pref}}$ given $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})$. Mathematically, this is expressed as:

$$\begin{aligned}
&\nabla_{\mathbf{z}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}}), \mathbf{x}_{\text{pref}}) \\
&\quad \approx \nabla_{\mathbf{z}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})).
\end{aligned}$$

This simplification allows us to express the conditional score function:

$$\begin{aligned}
&\nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}|\mathbf{x}_{\text{pref}}, \mathbf{z}_t) \\
&\approx \nabla_{\mathbf{z}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})) \\
&= -\nabla_{\mathbf{z}_t} \ell_{\mathbf{y}}(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})
\end{aligned}$$

where $\ell_{\mathbf{y}}(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})$ is the cross-entropy loss. With this, plug-and-play guidance simply requires a classifier within the sentence-T5 latent space. We employ a linear probe (i.e. logistic regression) in our experiments (see Appendix C for additional details). We find that semantic diffusion enables effective control with surprisingly simple classifiers.

Song et al. (2023) observed that the MMSE estimate, $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}})$, introduced approximation errors in the conditional score estimate. They propose sampling around the MMSE estimate

$$\hat{\mathbf{x}}^{(i)} \sim \mathcal{N}(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t, \mathbf{x}_{\text{pref}}), \sigma_t^2/\alpha_t^2 \mathbf{I}).$$

The sampling distribution has large variance early in the sampling process when the DPS estimate is

		C4			OpenWebText		
	Prefix Guidance (w)	MAUVE ↑	OLMo Ppl ↓	Div ↑	MAUVE ↑	OLMo Ppl ↓	Div ↑
Reference	-		19.2	58.4	-	17.2	57.6
GPT-2_{Large}	-	$83.9_{.3}$	116.3	50.3	$88.2_{.3}$	17.6	49.2
DGLM	1.0	$84.0_{.4}$	30.1	50.8	$78.6_{.4}$	22.9	50.2
DGLM	1.5	$85.6_{.4}$	23.0	52.5	$82.8_{.3}$	17.1	51.4
DGLM	2.0	$84.8_{.8}$	21.4	53.3	$83.1_{.3}$	15.4	52.1
DGLM	2.5	$84.8_{.1}$	20.2	54.0	$83.7_{.4}$	15.0	52.4
DGLM	3.0	$86.6_{.2}$	19.8	54.0	$84.5_{.4}$	14.7	52.5
DGLM	5.0	$85.6_{.4}$	19.4	53.9	$84.0_{.3}$	14.2	52.6

Table 1: Language generation evaluation. For the MAUVE score, we report the standard error of the mean over 5 random seeds.

uncertain and converges to the DPS point estimate at the end of the sampling process. They use a Monte-Carlo approach to approximate the guidance with the logmeanexp operation. Adapting this, we compute the guidance term as:

$$-\nabla_{\mathbf{z}_t} \log(\frac{1}{n}\sum_{i}^{n} \exp(\ell_{\mathbf{y}}(\hat{\mathbf{x}}^{(i)}))).$$

Early in the sampling process, this steers generation towards a region of low loss within the latent space. With $n = 32$, using the Monte-Carlo estimate incurs negligible overhead, requiring only 32 logistic regression evaluations.

4 Datasets and Metrics

Datasets. We extract a subset of 10 million instances from C4 (Raffel et al., 2019) to pre-train DGLM. This represents only 2.5% of C4 and scaling the pre-training corpus would likely be fruitful. We follow Geiping and Goldstein (2023) and filter out uncompressible text to improve quality. If the number of GPT-2 tokens is more than $t = 0.3$ times the raw number of characters, we drop it from the dataset. This removes instances consisting of, for instance, long HTML strings or markdown code.

To evaluate the language generation capabilities of our DGLM, we extract 5000 random validation instances from C4 (Raffel et al., 2019) and OpenWebText (Gokaslan and Cohen, 2019). We condition the network on the first 32 tokens and generate a 32 token continuation. For our toxicity mitigation experiments, we train our logistic regression model on the Jigsaw Unintended Bias dataset (cjadams, 2019) and evaluate the effectiveness of toxicity mitigation experiments using 5,000 neutral promps from RealToxicityPrompts (Gehman et al.). For our sentiment control experiments, we utilize Amazon Polarity [2] and SST-2 (Socher et al., 2013) to train a sentiment classifier, and perform sentiment control using 5,000 neutral prompts from OpenWebText.

Metrics. We evaluate the fluency of text by measuring its perplexity with the open-source OLMo-1B[3] language model. We also report MAUVE scores (Pillutla et al., 2021), a text generation metric that measures the similarity of generated text with that of reference text using divergence frontiers. To get embeddings for MAUVE, we follow the advice of He et al. (2022) and utilize ELECTRA-large (Clark et al., 2020). To evaluate generation diversity, we use the metric introduced by Su et al. (2022): $\textbf{Div} = \prod_{n=2}^{4} \frac{|\text{unique n-grams}(\{\mathbf{w}_i\})|}{|\text{total n-grams}(\{\mathbf{w}_i\})|}$ where $\{\mathbf{w}_i\}$ is a set of generated samples.

For the guidance tasks, we generate 25 samples per prompt. We report the OLMo-1B perplexity of the continuations to evaluate the fluency of the generations. We follow prior work and measure the average number of unique 3-grams, denoted Dist-3, in each set of continuations to quantify generation diversity. Along with ensuring that guidance does not degrade language quality or sacrifice diversity, we measure the adherence to the guidance conditions. Following prior work (Deng and Raffel, 2023; Liu et al., 2021), we use the Perspective API to measure the toxicity of generated text. Because Pozzobon et al. (2023) found that the Perspective API changes significantly over time, we re-score the released generations for all of the baselines with the current version of the API. We measure the *average max toxicity* across 25 generations and the *toxicity rate*, defined as the empirical odds of at least 1 of 25 continuations being classified as toxic.

To evaluate sentiment, we utilize RoBERTa-

[2] https://huggingface.co/datasets/amazon_polarity

[3] https://huggingface.co/allenai/OLMo-1B

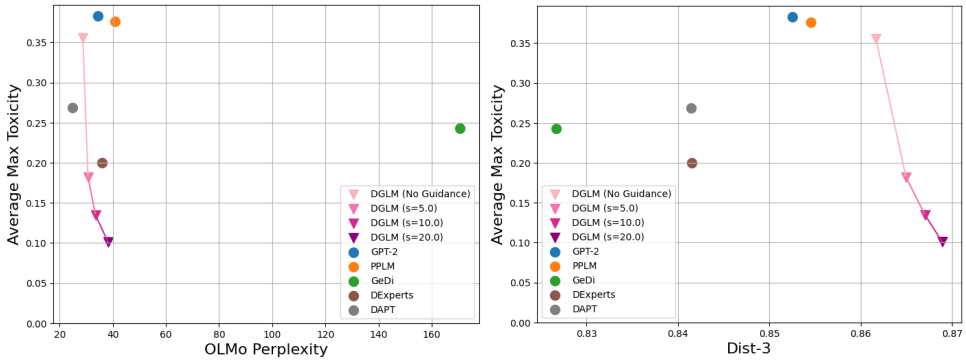

Figure 5: Effect of mitigating toxicity with increasing guidance weights. Increasing guidance reduces toxicity with minimal loss of fluency.

Large[4] (Liu et al., 2020) fine-tuned on sentiment classification across diverse domains as well as the fine-tuned DistilBERT model (Sanh et al., 2019) used by prior work.

5 Experimental Results

Language Generation. We validate the effectiveness of our framework on open-ended language generation in Table 1 without any plug and play control. We observe that our method achieves *strong* language generation results, matching or surpassing the reference perplexity with sufficient classifier-free guidance strength. We observe that DGLM leads to consistently *more diverse* generations than the auto-regressive baseline across both datasets. We observed that a handful of very high perplexity samples skews the GPT-2 baseline's perplexity on C4. However, DGLM also achieves stronger MAUVE scores on that dataset.

We examine the impact of Gaussian noise augmentation in Table 2. As an additional metric, we re-embed the generated text with Sentence-T5 and compute the cosine similarity with the proposal embedding[5]. We observe that the Gaussian noise augmentation enables the network to smoothly interpolate between auto-regressive generation (low perplexity but poor diversity) and diffusion-guided generation (higher perplexity and diversity). We observe that lower levels of noise montonically improve the decoders adherence to the proposal.

Plug-and-Play Control. We utilize DGLM to avoid generating toxic language. We show quantitative results in Figure 5 and Figure 7. Qualitative examples are presented in Table 9. Plug-and-play guidance with a linear probe effectively mitigates toxicity with negligible trade-offs in fluency. We simultaneously achieve lower perplexity, lower toxicity, and higher diversity than all baselines.

		C4		
	Noise (σ_t^2)	S-T5 Sim ↑	OLMo Ppl ↓	Div ↑
Reference	-	35.7	19.2	58.4
DGLM	1.0	36.7	17.3	45.9
	0.8	45.6	21.8	47.1
	0.6	50.9	22.9	48.6
	0.4	54.6	26.1	49.8
	0.2	56.8	28.1	50.3
	0.05	58.5	30.1	50.8
	0.0	59.1	30.7	51.4

Table 2: Impact of Gaussian noise augmentation. $\sigma_t^2 = 1.0$ corresponds to Gaussian noise and $\sigma_t^2 = 0.0$ corresponds to the clean proposal.

We also employ DGLM to steer the sentiment of generated text. We present results for negative sentiment in Figure 6 and positive sentiment in Figure 8. We observe that our method is similarly effective in this setting, decreasing (or increasing) sentiment with no loss of fluency and minimal loss of diversity for modest guidance values.

Compositional Control. We present qualitative results from composing multiple attribute classifiers with DGLM. We fine-tune an additional logistic regression model on the AG News topic classification dataset. We then sum the losses for the sentiment and topic classification classifier to guide generation. We find that DGLM successfully enables compositional control and present qualitative examples in Table 3 (additional examples are in Table 10). We leave rigorous evaluations of the compositionality of DGLM to future work.

[4] https://huggingface.co/siebert/sentiment-roberta-large-english

[5] We follow Zhang* et al. (2020) and rescale the cosine similarity with a baseline computed between random dataset samples.

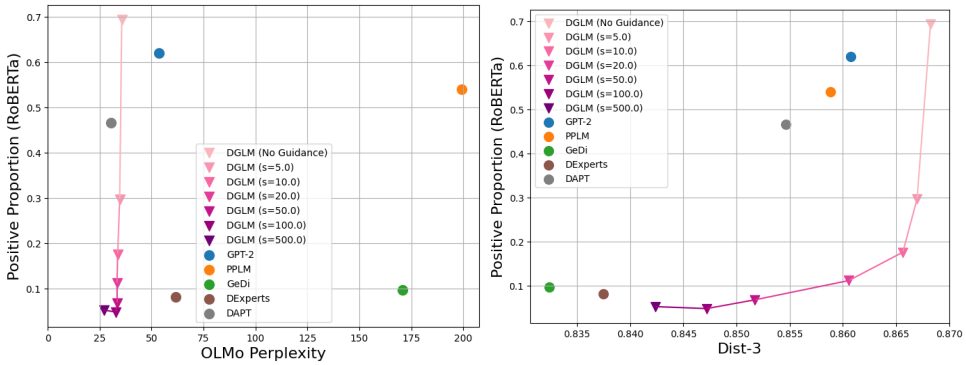

Figure 6: Effect of guiding generations towards negative sentiment with increasing guidance weights. Increasing guidance improves alignment with the target sentiment while sacrificing diversity.

Topic	Sentiment	Prefix	Continuation
Sci/ Tech	Negative	Therefore, we will not	provide a technical review of the software, including its capabilities, nor will we provide you with any reports or comments regarding the accuracy of information.
Sports	Positive	Other than that,	I think we really did a great job of letting the fans know how it felt to see them come out in record numbers for an 82 game season.

Table 3: Language generated by controlling two attributes simultaneously.

Decoding Overhead. Plug-and-play methods for auto-regressive generation often introduce overhead at each decoding step. For example, DExperts employs auxiliary language models that work alongside the primary model. In contrast, DGLM incurs a one-time cost for generating the semantic proposal, which is then amortized across subsequent decoding steps. We therefore compute runtimes across a range of generation lengths. We report the relative increase in runtime compared to the original GPT-2 model for each method (baseline data from Liu et al. (2021)) in Table 4. As seen in the table, DGLM incurs a large cost for short sequences but has reduced overhead compared to prior methods with modest generation lengths.

Method	Relative Runtime
GPT-2	1.0x
GeDi	2.9x
DeXperts (large)	3.6x
PPLM	270.1x
DGLM (32 tokens)	7.4x
DGLM (64 tokens)	4.4x
DGLM (128 tokens)	2.6x
DGLM (256 tokens)	1.7x

Table 4: Relative runtime compared to GPT-2.

6 Related Work

Fine-tuning. Continual pre-training on text from some target domain (domain-adaptive pretraining or DAPT) is an effective technique for controlling attributes in generated text (Gururangan et al., 2020). Lu et al. (2022) optimize a reward function by fine-tuning an LM with control tokens for different reward quantiles. Reinforcement Learning with Human Feedback (RLHF) involves training a reward model on human preference data that is then used to fine-tune the LM (Wu et al., 2023; Ouyang et al., 2022). Jang et al. (2023) train multiple personalized RLFH models and show that these personalized models can be used alone or in conjunction with one another to produce text with desired attributes.

Guided Generation. Finetuning LMs is expensive and therefore to reduce cost, Dathathri et al. (2019) proposed Plug and Play Language Model (PPLM), a method that used light-weight classifiers to guide frozen language models during text generation. Similarly, FUDGE (Yang and Klein, 2021) trains classifiers on partial sequences to predict whether a particular attribute is satisfied and updates the output probability distribution accordingly. Instead of using a classifier, GeDi (Krause et al., 2021) trains a small class-conditional language model to act as a discriminator and guide

the language generation. Similarly, DeXperts (Liu et al., 2021) trains experts and anti-experts by fine-tuning small language models, and using these experts to guide generation. Reward-Augmented Decoding (RAD) (Deng and Raffel, 2023) trains a reward model to score generations and adjust logit probabilities to promote high-reward tokens.

7 Conclusion

We present Diffusion Guided Language Modeling (DGLM), a powerful integration of auto-regression and diffusion that enables versatile attribute-guided text generation with lightweight classifiers. The diffusion model generates controllable semantic proposals that guide the language decoder. Extending DGLM to control an unseen attribute only requires learning a single logistic regression model. Experimental results show that DGLM significantly outperforms prior plug-and-play methods, opening avenues for building highly adaptable LMs with user-controllable behavior.

8 Limitations

While DGLM demonstrates strong capabilities for guided text generation, we acknowledge important limitations. First, like any system that controls text attributes, it risks potential misuse to steer language in harmful directions. Researchers and practitioners should carefully evaluate generation systems to mitigate these risks.

In addition, DGLM currently has slower inference speed than some plug-and-play baselines when generating short texts (<32 tokens). We expect advances in accelerating diffusion models and distilling diffusion steps will help address this limitation in future work.

More broadly, while DGLM outperforms recent methods, there is still substantial room for improvement in controllable text generation. The framework currently utilizes simple linear classifiers that may not robustly capture complex attributes. Extending DGLM to complex attributes may require more complex classifiers. We hope our work sparks further research towards reliable and beneficial guided language models.

Acknowledgements

This research is supported by grants from the National Science Foundation NSF (IIS-2107161, and IIS-1724282, HDR-2118310), the Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875), DARPA, arXiv, LinkedIn, and the New York Presbyterian Hospital.

References

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. 2022. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. 2023. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*.

inversion Jeffrey Sorensen Lucas Dixon Lucy Vasserman nithum cjadams, Daniel Borkan. 2019. Jigsaw unintended bias in toxicity classification.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. 2023. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models.

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*.

Ishaan Gulrajani and Tatsunori B Hashimoto. 2023. Likelihood-based diffusion language models. *arXiv preprint arXiv:2305.18619*.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Tianxing He, Jingyu Zhang, Tianle Wang, Sachin Kumar, Kyunghyun Cho, James Glass, and Yulia Tsvetkov. 2022. On the blind spots of model-based evaluation metrics for text generation. *arXiv preprint arXiv:2212.10020*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. 2022. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. 2023. simple diffusion: End-to-end diffusion for high resolution images.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. 2021. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707.

Diederik P Kingma and Ruiqi Gao. 2023. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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 *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952.

Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. 2020. Multi-agent communication meets natural language: Synergies between functional and structural language learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7663–7674.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343.

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 *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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Seo Shekhtman, and Kilian Q Weinberger. 2023. Latent diffusion for language generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. Quark: Controllable text generation with reinforced unlearning. *Advances*

in neural information processing systems, 35:27591–27609.

John X Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. 2023. Text embeddings reveal (almost) as much as text. *arXiv preprint arXiv:2310.06816*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874.

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR.

Michael Noukhovitch, Samuel Lavoie, Florian Strub, and Aaron Courville. 2023. Language model alignment with elastic reset. In *Neural Information Processing Systems*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

William Peebles and Saining Xie. 2022. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Luiza Amador Pozzobon, Beyza Ermis, Patrick Lewis, and Sara Hooker. 2023. On the challenges of using black-box APIs for toxicity evaluation in research. In *ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Colin Raffel, Noam 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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-resolution image synthesis with latent diffusion models.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. 2022. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494. Curran Associates, Inc.

Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. 2023. Loss-guided diffusion models for plug-and-play controllable generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32483–32498. PMLR.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022. A contrastive framework for neural text generation. *arXiv preprint arXiv:2202.06417*.

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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

Kevin Yang and Dan Klein. 2021. 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.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. 2023. Planner: Generating diversified paragraph via latent language diffusion model. *arXiv preprint arXiv:2306.02531*.

A Additional Figures

We present toxicity mitigation results with the *Toxic Rate* metric in Figure 7. We present plug-and-play results with positive sentiment guidance in Figure 8. We present the sentiment guidance results with the DistilBERT classifier used in prior work in Figure 9 and Figure 10.

B Numerical Results

We provide the numerical results for our toxicity mitigation and sentiment control experiments in Table 5 and Table 6.

For the baseline methods, we observed a handful of extremely high perplexity generations (e.g. >1e4) that significantly increase the average perplexity. Prior work typically filters out these instances when computing the average perplexity[6]. We did not observe any such high perplexity continuations for our method. We therefore do *not* perform this filtering for our method.

C Implementation Details

We train all of the models in this work on a single NVidia A6000 GPU.

Transformer Implementation. We use different configurations of the same transformer architecture for the prompt generator and the diffusion network. We utilize a pre-normalization transformer (Vaswani et al., 2017; Xiong et al., 2020) with RMSNorm (Zhang and Sennrich, 2019) and SwiGLU activations (Shazeer, 2020). We condition the transformer on the level of noise by mapping α_t to a sinusoidal positional embedding (Vaswani et al., 2017) and pass it through an MLP with a single hidden layer to obtain a time embedding. We apply adaptive RMSNorm conditioned on this time embedding before the feedforward layers and attention layer (Peebles and Xie, 2022). We utilize query-key RMSNorm (Dehghani et al., 2023) for the self-attention mechanisms because it has been shown to improve stability.

Diffusion Network. We employ the v-parameterization and minimize:

$$\mathcal{L}_{\mathbf{v}}(\mathbf{x}) = \mathbb{E}_{t,\mathbf{x},\boldsymbol{\epsilon}}[w(\lambda_t) \cdot \|\hat{\mathbf{v}}_\theta(\mathbf{z}_t; \lambda) - \mathbf{v}_t\|_2^2].$$

To set the weighting function, we followed the advice of (Karras et al., 2022) and parameterized it with a log-normal distribution based on the noise levels where the model was best able to minimize the loss. This led us to set $w(\lambda_t) = \mathcal{N}(\lambda_t; 0, 2.4)$. Consistent with past work (Balaji et al., 2022), we observed that increasing weights at high noise levels improved the alignment of generations with the conditioning information. For our final weighting function, we therefore used a fat-tailed Cauchy distribution for the left half of the distribution and a normal distribution for the right half. This gives us

$$w(\lambda_t) = \begin{cases} \frac{1}{Z_c}\text{Cauchy}(\lambda_t; 0, 2.4) & \text{if } \lambda_t < 0 \\ \frac{1}{Z_n}\mathcal{N}(\lambda_t; 0, 2.4) & \text{if } \lambda_t \geq 0 \end{cases}$$

where Z_c and Z_n are normalization constants such that the density of each distribution at 0 is re-scaled to 1. For training, we utilize the adaptive noise scheduler introduced by Kingma and Gao (2023) to reduce the variance of the loss estimate.

Sampling Configuration. We use the stochastic DDPM sampler with 50 sampling steps with the cosine noise schedule (Nichol and Dhariwal, 2021). We follow Hoogeboom et al. (2023) and set the variance for the DDPM sampler to a log-scale interpolation between the upper and lower bounds of the variance from Ho et al. (2020): $\sigma^2 = \exp(v \log(\sigma^2_{\text{max}}) + (1 - v) \log(\sigma^2_{\text{min}}))$ with $v = 0.2$. We did not explore this choice in detail and further exploration of sampling configurations would likely improve performance.

Logistic Regression Classifiers. We train logistic regression models with sci-kit learn. We utilize the default L-BFGS solver with L2 regularization of 1e-3. We use balanced class weights for the toxicity classifier due to the class imbalance in the toxicity dataset. Our toxicity and sentiment classifiers achieve an Area Under the Receiver Operating Curve of 83.7 and 95.7, respectively.

D Additional Composition Results

As specified in the main paper, DGLM is naturally suited for simultaneously controlling multiple attributes. Table D presents additional qualitative results for compositional control. From the table, we see that the instances satisfies both control attributes.

[6] See here for an example.

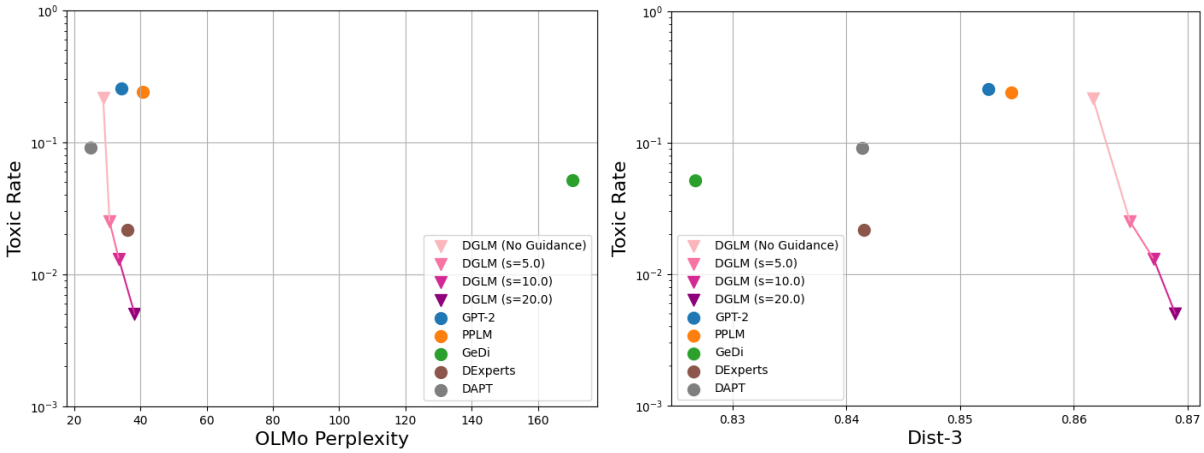

Figure 7: Effect of plug-and-play toxicity mitigation with increasing guidance weights. We observe that increasing guidance reduces toxicity at the cost of language fluency.

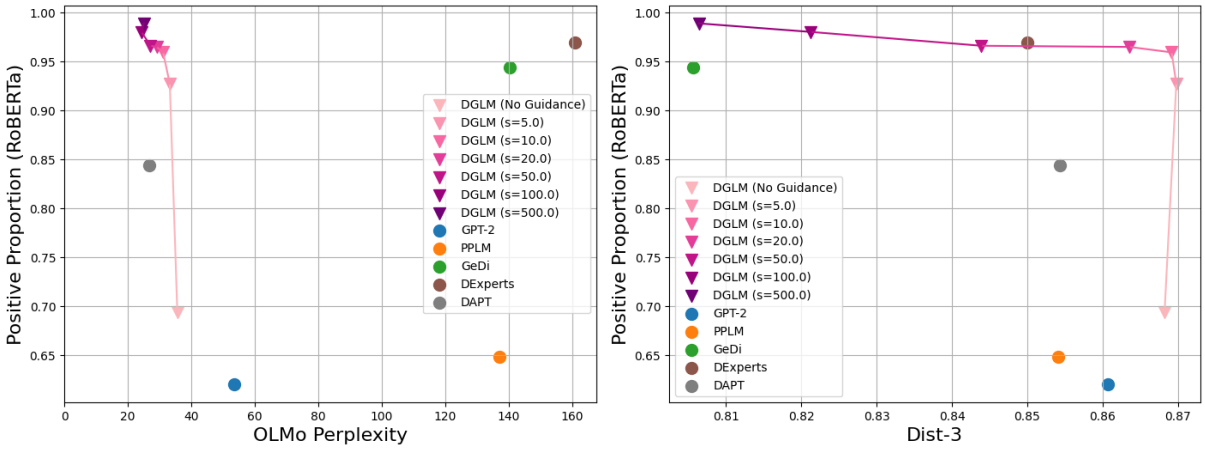

Figure 8: Effect of guiding generations towards positive sentiment with increasing guidance weights. The left plot shows the impact on language perplexity and the right plot shows the impact on language diversity.

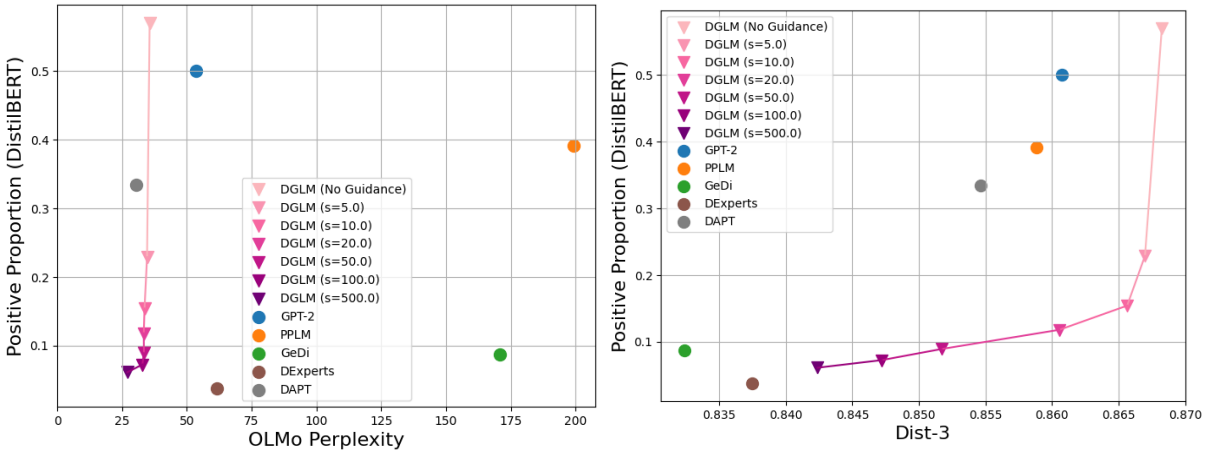

Figure 9: Effect of guiding generations towards negative sentiment with increasing guidance weights. The left plot shows the impact on language perplexity and the right plot shows the impact on language diversity.

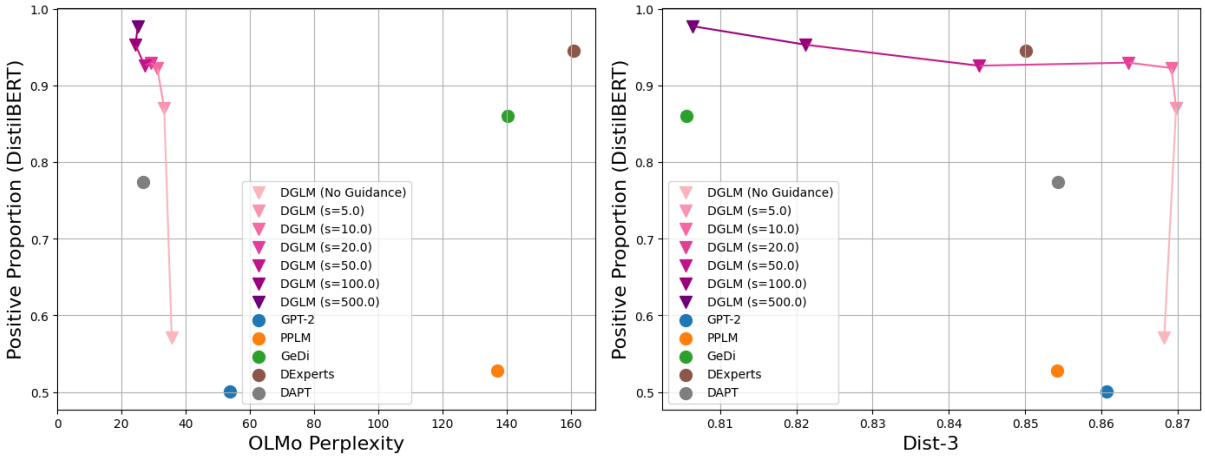

Figure 10: Effect of guiding generations towards positive sentiment with increasing guidance weights. The left plot shows the impact on language perplexity and the right plot shows the impact on language diversity.

Method	Avg. Max Toxicity ↓	Toxic Rate ↓	OLMO Ppl ↓	Dist-3 ↑
GPT-2	0.383	0.254	34.4	0.853
DAPT	0.269	0.091	24.9	0.841
PPLM	0.376	0.240	40.8	0.855
GeDi	0.243	0.051	170.5	0.827
DExperts	0.200	0.022	36.0	0.842
DGLM (No Guidance)	0.355	0.218	28.8	0.862
DGLM (s=5.0)	0.182	0.025	30.7	0.865
DGLM (s=10.0)	0.135	0.013	33.5	0.867
DGLM (s=20.0)	0.101	0.005	38.2	0.869

Table 5: Toxicity Mitigation Results.

Target Sentiment: Positive			
Method	Positive Prop. (RoBERTa) ↑	OLMO Ppl ↓	Dist-3 ↑
GPT-2	0.621	53.6	0.861
DAPT	0.844	26.8	0.854
PPLM	0.649	137.1	0.854
GeDi	0.944	140.4	0.806
DExperts	0.969	160.8	0.850
DGLM (No Guidance)	0.694	35.6	0.868
DGLM (Guidance 5.0)	0.927	33.2	0.870
DGLM (Guidance 10.0)	0.959	31.1	0.869
DGLM (Guidance 20.0)	0.965	29.1	0.864
DGLM (Guidance 50.0)	0.966	27.2	0.844
DGLM (Guidance 100.0)	0.980	24.3	0.821
DGLM (Guidance 500.0)	0.989	25.0	0.806
Target Sentiment: Negative			
Method	Positive Prop. (RoBERTa) ↓	OLMO Ppl ↓	Dist-3 ↑
GPT-2	0.621	53.6	0.861
DAPT	0.466	30.3	0.855
PPLM	0.540	199.1	0.859
GeDi	0.097	170.7	0.832
DExperts	0.082	61.7	0.837
DGLM (No Guidance)	0.694	35.6	0.868
DGLM (Guidance 5.0)	0.297	34.6	0.867
DGLM (Guidance 10.0)	0.176	33.6	0.866
DGLM (Guidance 20.0)	0.112	33.3	0.861
DGLM (Guidance 50.0)	0.068	33.4	0.852
DGLM (Guidance 100.0)	0.048	32.8	0.847
DGLM (Guidance 500.0)	0.053	27.0	0.842

Table 6: Sentiment Control Results.

Table 7: Implementation details for auto-regressive pre-training stage.

Prompt Generator Architecture	Pre-Activation Transformer (Vaswani et al., 2017; Xiong et al., 2020)
Soft Prompt Tokens	8
Transformer Layers	6
Transformer Dimension	768
Self-Attention Heads	12
Activation Function	SwiGLU (Shazeer, 2020)
Normalization Layer	Adaptive RMSNorm (Zhang and Sennrich, 2019; Peebles and Xie, 2022)
Max Seq Length	96
Optimizer	AdamW (Loshchilov and Hutter, 2019)
Learning Rate	5e-6
(β_1, β_2)	(0.9, 0.99)
Batch Size	64
Warmup Steps	5000
Learning Rate Schedule	Cosine Decay
Weight Decay	.02
Gradient Clipping	1.0
Batch Size	64
Augmentation Noise Schedule	Scaled Cosine (s=3.0) (Hoogeboom et al., 2023)
Training Steps	250k

Table 8: Implementation details for diffusion training.

Diffusion Architecture	Pre-Activation Transformer (Vaswani et al., 2017; Xiong et al., 2020)
Input Tokens	64
Transformer Layers	12
Transformer Dimension	768
Transformer Output Projection Dimension	96
Final Feature Vector Dimension	96*64
Self-Attention Heads	12
Activation Function	SwiGLU (Shazeer, 2020)
Normalization Layer	Adaptive RMSNorm (Zhang and Sennrich, 2019; Peebles and Xie, 2022)
Optimizer	AdamW (Loshchilov and Hutter, 2019)
Learning Rate	1e-3
(β_1, β_2)	(0.9, 0.999)
Batch Size	256
Warmup Steps	1000
Learning Rate Schedule	Cosine Decay
Weight Decay	.1
Gradient Clipping	1.0
Batch Size	256
Training Noise Schedule	Adaptive Kingma and Gao (2023)
Sampling Noise Schedule	Cosine Nichol and Dhariwal (2021)
Training Steps	250k

Guidance	Prefix	Continuation
Low Toxicity	She made out with him, took sexy pictures in a photo booth, and watched him	have fun with his friends. She shared the photos on her private Facebook page, where a family...
Positive Sentiment	Cycle, published by the CTC, is running	its 10th edition and it is getting better every time I see the contents! It's also very...
Negative Sentiment	Cycle, published by the CTC, is running	its 'news' section, with no substance at all and zero interest in the subject it...

Table 9: Qualitative examples of guided generation.

Topic	Sentiment	Prefix	Continuation
Sports	Negative	The person that makes such decisions	and coaches the team, is really just a bad sports person. That's all there ever was to it. I think we are going down...
Business	Negative	I realized this 10 years	ago when I started running my own business and it was very clear to me that Sunland was really not making any money...
Sci/ Tech	Positive	And that's pretty much the	best part about this site – you can get all sorts of great technical info with just a couple clicks...

Table 10: Additional Qualitative examples of compositional control.