Controlled Text Generation of DLLMs with Efficient Classifier Guidance

Zhuo Cao[*1], **Xuanyi Xie**[*1], **Qingyan Wei**[2], **Jiawang Zhao**[1]

[1]Institute of Interdisciplinary Information Sciences, Tsinghua University
[2]School of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China
caozhuo23@mails.tsinghua.edu.cn, xie-xy23@mails.tsinghua.edu.cn, florawei0506@gmail.com,
zhao-jw25@mails.tsinghua.edu.cn

Abstract

Recent advancements in large language models (LLMs) have significantly enhanced many aspects of research and daily lives. While LLMs are inherently powerful, their performance can be further improved through post-training tailored to specific user needs. For individual users who are unable to directly finetune base models due to resource constraints, classifier guidance offers a more cost-effective solution for controlled text generation. However, traditional ways of applying classifier guidance (CFG) dose not transfer well to Diffusion Large Language Models (DLLMs), which would suffer from instability, criterion mismatch and slow inference. In this work, we propose an efficient approach to utilize and adapt classifier guidance on DLLMs, substantially reducing training and inference time while delivering notable improvements in task performance.

1 Introduction

Recently, Diffusion Large Language Models (DLLMs) have made significant progress in multiple aspects. (Nie et al. 2025)(Ye et al. 2025) The ability of DLLMs to generate tokens in parallel via an iterative unmasking mechanism provides them with inherent potential advantages, namely reduced inference latency and superior bidirectional context awareness.(Li et al. 2025c)

Among all the research directions, controlled text generation is scarcely mentioned in DLLMs. However, it is worth exploring, as its bidirectional context awareness nature should enable fine-grained control over the generation process, revealing the possibility of efficient personalization.(Li et al. 2025c)

Classifier guidance has already been well studied in the field of auto-regressive models, e.g., PPLM (Dathathri et al. 2020), FUDGE (Zhao, Wang, and Kolar 2022).Existing works that consider applying classifier guidance on DLLMs either focus on some very specific tasks, like Arg-LLaDA (Li et al. 2025a) and DINGO (Suresh et al. 2025), or are too heavy and complicated to be generally applied, such as CtrlDiff (Huang and Tang 2025). And most importantly, the dramatic speed cut of inference when a classifier guidance (CFG) is introduced does not seem to be sufficiently discussed in these works.

In this paper, we propose an efficient and general classifier guidance method for DLLMs that not only saves training cost but also makes inference gigantically more rapid than vanilla CFG via three main methods:

- **Stabilized Guidance Form.** We modify the post-transformation of CFG probabilities to amplify the internal differences, especially within high-confidence regions.
- **Implicit Prompting.** To bridge the criterion gap between human and trained classifier, we give larger guidance weight in the beginning steps of generation, thus implicitly helping later iterations generate proper tokens.
- **Early-stopping.** Once the predicted probability of the intermediate result is fairly high, we stop invoking CFG in later steps and continue with original denoising process.

2 Related Work

Discrete Diffusion Language Models

We follow the notations in (Nie et al. 2025). First let us clarify the goal of all language models. Let the true data distribution of some corpus be $p_{\text{data}}(\cdot)$, and we want to optimize a language model $p_\theta(\cdot)$ to get close to this true distribution. The goal can be measured by the log-likelihood of the model distribution:

$$\max_\theta \mathbb{E}_{p_{\text{data}}(x)}(\log p_\theta(x)) \tag{1}$$

In discrete diffusion language models, a forward process and a reverse process are defined. Let $t \in (0, 1)$ be the diffusion timestamp. x_t denotes the token sequence at diffusion timestamp t. In the forward process, we randomly mask the tokens in x_0 until all the tokens are masked in x_1. In x_t, every token is masked with probability t, or remain unmasked with probability $(1 - t)$. The reverse process tries to predict the tokens on masked positions and recover the original x_0.

The core of discrete diffusion language models is a mask predictor, a model $p_\theta(\cdot|x_t)$ which takes x_t as input and predicts all masked tokens (denoted as [MASK]) simultaneously. It is trained using the following cross-entropy loss

[*]These authors contributed equally.

function:

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = \text{[MASK]}]\log p_\theta(x_0^i|x_t)\right], \tag{2}$$

where t is sampled from $[0,1]$, x_0 is sampled from the training dataset, and x_t is sampled from the forward process.

It is proved that Eq. (2) is an an upper bound on the negative log-likelihood of the model distribution in Eq. (1).(Shi et al. 2024) Therefore, it is reasonable to optimize the mask predictor by Eq. (2) to get a better modeling of the original data distribution.

$$-\mathbb{E}_{p_{\text{data}}(x_0)}\left[\log p_\theta(x_0)\right] \leq \mathcal{L}(\theta), \tag{3}$$

During inference time, the mask predictor predicts all masked tokens simultaneously, and unmask some of the tokens. Once a token is unmasked, it remains unchanged until the end of the reverse process. The unmasking strategy can be various, including random unmasking and highest confidence unmasking. Also, for diffusion language models that are finetuned with paired prompt and response data for better instruction following, we can divide the sequence into several blocks and generate them from left to right, called semi-autoregressive generation. We still apply the mask predictor within every block. It is proved through experiments that using highest confidence unmasking strategy and semi-autoregressive generation gives the best generation quality for instructed diffusion language models.

Controlled Text Generation

With the rapid growth of Large Language Models, text generation quality has improved significantly. However, in real-world scenarios, LLMs often face more complex and stricter requirements. In some specific domains, generated content must not only avoid being misleading or discriminatory, but also precisely meet specific conditions and user expectations, such as following a strict format (e.g. json, LaTeX), mimicking a certain writing style, or keeping a specific sentiment and standing point. These needs have driven the advancement of Controlled Text Generation (CTG), which aims to ensure that the output is both high-quality and tailored to the unique demands of various applications.

The methods of controlled text generation can be mainly divided into two categories: training stage methods and inference stage methods. Training stage methods include retraining(He 2021), supervised fine-tuning(Zeldes et al. 2020), and reinforcement learning(Dai et al. 2023), etc.. Inference stage methods include prompt engineering(Lester, Al-Rfou, and Constant 2021), latent space manipulation(Liu et al. 2023) and decoding time intervention(Krause et al. 2020), etc..

While training stage methods have shown good controlling quality with no additional inference cost, they have the drawback of requiring extremely high training or finetuning cost. Also, this method is not transferrable, that is, we need to retrain the whole process if we use another base model. Prompt engineering is training-free, but it is dirty and domain specific. It can also be hard to summarize a precise prompt for some domains.

Classifier Guidance

Classifier guidance is a common technique in controlled text generation, and has proved to be effective for auto-regressive models. This method has several advantages: First, its training cost does not depend on the size of the base model, but on the design of the classifier, so it usually requires much less training resources than other methods that directly modifies the original model weight, like SFT and LoRA (Hu et al. 2021). Second, a classifier can be easily transferred among different base models as long as they share the same tokenizer. There is no need to retrain the classifier if the base model's parameters are updated. Third, they can be stacked together easily and perform multi-attribute control. As a result, classifier guidance is an ideal light-weight method for controlled text generation.

PPLM (Plug and Play Language Model) (Dathathri et al. 2020) is one of the earliest approaches for controlled text generation through latent space manipulation. It works by combining a pre-trained language model with attribute classifiers, which guide the output by steering hidden layer activations using gradients from those classifiers. This enables control over attributes like topic or sentiment without changing the underlying language model itself. While the method sometimes sacrifices fluency, its flexibility allows for the use of multiple controllers to manage more complex aspects of generation.

At each generation step t, PPLM modifies the direction of the past activations H_t to influence the language model's output: (Dathathri et al. 2020)

$$\Delta H_t = \Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)}{\|\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)\|^\gamma}$$

where α is the step size and γ is the normalization coefficient. After updating ΔH_t, the language model performs a forward pass to obtain the updated logits $\tilde{o}_{t+1}$:

$$\tilde{o}_{t+1}, H_{t+1} = \text{LM}(x_t, \tilde{H}_t), \quad \tilde{H}_t = H_t + \Delta H_t$$

These logits define a new probability distribution $\tilde{p}_{t+1}$, from which the next token is sampled.

FUDGE (Future Discriminators for Generation) (Zhao, Wang, and Kolar 2022) provides a simpler and more efficient alternative to PPLM by adjusting the probability distribution in real time during text generation. It estimates the likelihood that the ongoing sequence matches a desired attribute and then modifies the logits to encourage that attribute. In practice, FUDGE treats text generation as $P(x_i \mid x_{1:i-1})$ and refines it via Bayesian factorization.

$$P(x_i \mid x_{1:i-1}, a) \propto P(a \mid x_{1:i})\, P(x_i \mid x_{1:i-1}),$$

where $P(a \mid x_{1:i})$ is modeled by a classifier.

There have also been some research on using classifier guidance approach for controlled text generation of DLLMs. For example, CtrlDiff (Huang and Tang 2025)implements a guidance mechanism for discrete diffusion models and it works well on DLLMs. However, it is too complicated in math explanation, not as clear and beautiful as FUDGE (Zhao, Wang, and Kolar 2022).

While there has been much exploration in classifier guided controlled text generation, this method still has the

problem of additional cost during inference time. When the vocabulary size is large, the time overhead of running classifiers can be 10 or even more times of running the base model. How to maintain a stable guidance, and balance control strength and output fluency is another problem for classifier guidance. To the best of our knowledge, our work is the first successful attempt on solving these problems in controlled text generation of DLLMs with classifier guidance.

3 Methods

Stabilize Guidance

We have implemented FUDGE (Zhao, Wang, and Kolar 2022) on DLLMs. Following the DLLM paradigm, at every inference step, the model first outputs logits for all the masked tokens inside the current block. Next, the classifier iterates all possible unmasking results and predicts the probability that this unmasking result will satisfy the desired attribute. Then we adjust the output logits of the base model with the predicted probability. Following FUDGE, the adjustment is:

$$\log P_\theta(x_i \mid x_{\text{unmasked}}) += \lambda \log P_\phi(a \mid x_{\text{unmasked}}, x_i),$$

where P_θ is the base model, x_{unmasked} is the already unmasked part. x_i is the next unmasked token(s), iterated among all possibilities. P_ϕ is the classifier, and a is the desired attribute. λ is the super-parameter for control strength.

In practice, we find this method does not provide a stable guidance. First, the guidance is actually determined by the difference between classifier outputs. In other words, it is the difference between $P_\phi(a \mid x_{\text{unmasked}}, x_i)$ and $P_\phi(a \mid x_{\text{unmasked}}, x_j)$ that impacts the choice of x_i or x_j. However, these two probabilities are both close to $P_\phi(a \mid x_{\text{unmasked}})$, as changing a few tokens does not impact the whole sentence much. Therefore, the classifier sometimes does not provide enough information to distinguish which is better for the desired attribute, especially when $P_\phi(a \mid x_{\text{unmasked}})$ is already large (say, larger than 0.99). As a result, the guidance put on is not significant enough if the original logit is relatively small.

To deal with this issue, we adjust the original guidance function to solve instability by amplifying the magnitude of those choices with higher classification confidence, without changing the basic idea of FUDGE (Zhao, Wang, and Kolar 2022). Now we transform the classifier output into log-odds. The adjustment becomes:

$$\log P_\theta(x_i \mid x_{\text{unmasked}}) += \lambda f(P_\phi(a \mid x_{\text{unmasked}}, x_i)),$$

where

$$f(x) = \log x - \log(1 - x).$$

The log-odds transformation amplifies the differences between candidates, especially when the classifier is already confident (i.e., probabilities close to 0 or 1). For instance, two candidates with probabilities 0.99 and 0.98 differ only marginally in $\log p$, but exhibit a much larger gap in log-odds, thereby producing a stronger adjustment to the base model's logits. Moreover, log-odds symmetrically handle both supportive and contradictory evidence, penalizing low-probability candidates more heavily and rewarding high-probability ones more strongly. As a result, the guidance signal becomes more sensitive to the classifier's relative preferences, preserving the essence of FUDGE (Zhao, Wang, and Kolar 2022) while improving and stabilizing its effectiveness in high-confidence regions.

Bridge Criterion Gap

The classification criterion of the trained classifier and human may differ a lot in certain cases, which would decrease the model's accuracy. More details of the criterion gap are in Appendix A. To bridge this gap, the model is forced to produce representative tokens at the beginning of the unmasking process, so that the final result is more likely to have consistent classification results between the human and the classifier.

In our method, we simply give larger guidance weight λ in the first few steps and then the original weight. This will generate an implicit prompt that helps later steps to generate proper tokens with higher probability.

Accelerate Inference

Directly applying FUDGE on DLLMs will cause a large inference overhead, especially when the vocabulary size is large. Suppose the generation length is L, block size is B, vocabulary size is V, and we unmask one token in one step. The total number of classifier invocations is

$$\frac{L}{B} \cdot \sum_{i=0}^{B-1} (B - i) \cdot |V| = O(LB|V|).$$

This is extremely large as $|V| \sim 10^5$ in large language models.

To deal with this issue, we first apply a top-k selection. We only calculate the classification probabilities of the top-k logits for every masked token. This is a reasonable approximation because, if a token receives a very small logit (not in top-k) from the base model, it is very likely that its logit will maintain in the backward part after adjustment and not likely to be selected. In practice, we set $k \sim 100$, and the total number of classifier invocations becomes $O(LBK)$, showing a significant speed-up.

In addition to the above improvement, we further reduce the inference overhead by introducing *selective guidance*. The intuition is that, if the already unmasked part of the sequence has a high classification confidence, applying classifier guidance at the current step will not significantly influence the outcome compared to standard conditional generation (e.g., classifier-free guidance).(The reason is similar to the explanation in section 3.1) In such cases, invoking the classifier is unnecessary and can be safely skipped.

Formally, we set a threshold τ for the classifier confidence. Only when $P_\phi(a \mid x_{\text{unmasked}}) < \tau$ do we invoke the classifier to compute the adjusted guidance score; otherwise, we proceed directly with a standard DLLM decoding step without classifier intervention. This approach avoids redundant classifier invocations in regions where the attribute likelihood is already sufficiently high. Also, we can expect

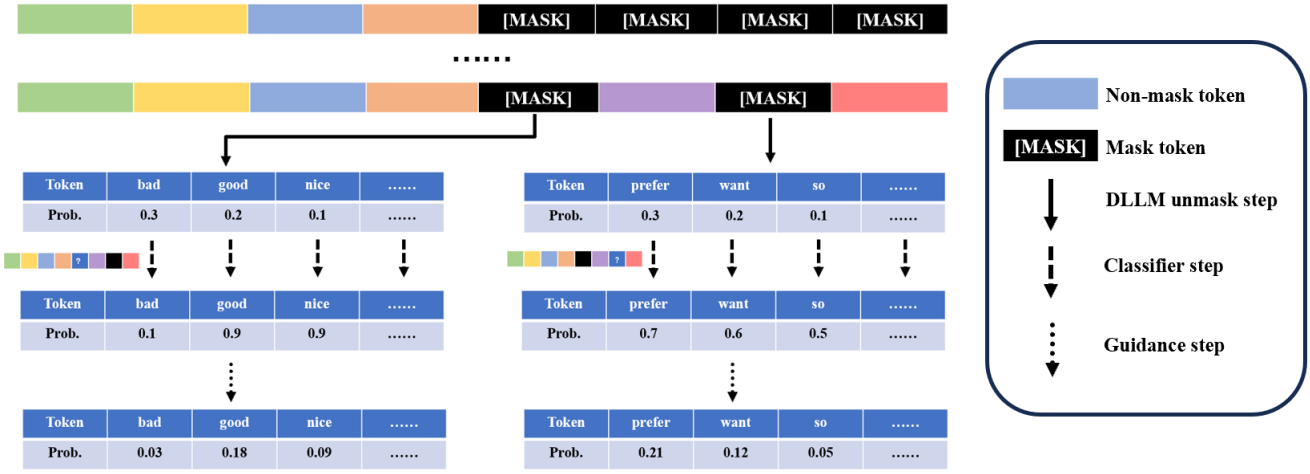

Figure 1: Method Overview. At every generation step, the base model outputs logits at every masked position. If the classifier's confidence on the current sentence is below threshold, it will calculate the probability that the candidate new sentences will satisfy the desired property, and adjust the logits.

that the next tokens the base model will generate to be align with the attribute we desire without the classifier guidance, as the base model should possess inherent concurrency to keep a certain sentiment or topic, etc.

4 Experiments

In our experiments, we adopted a BERT-based classifier architecture of ∼80M trainable parameters. More details on training classifier are listed in Appendix B.

Ablation Study

We compared the generation process of our method and vanilla FUDGE CFG on a test set containing 25 sentence completion tasks. The model needs to complete the sentence given the beginning words while keeping a certain sentiment (positive/negative).

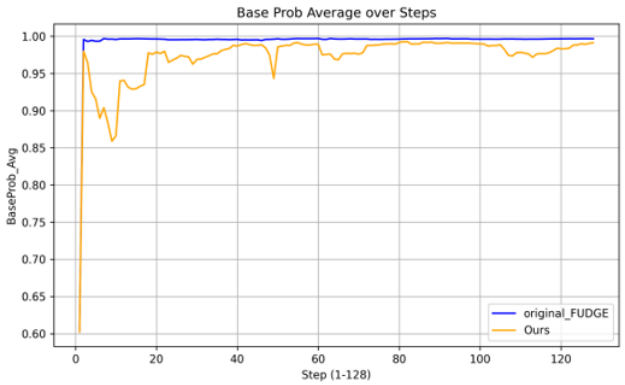

Figure 2: Average output probability of the classifier on the unmasked part w.r.t. generation step, which indicates that ours converges slower and provides more information.

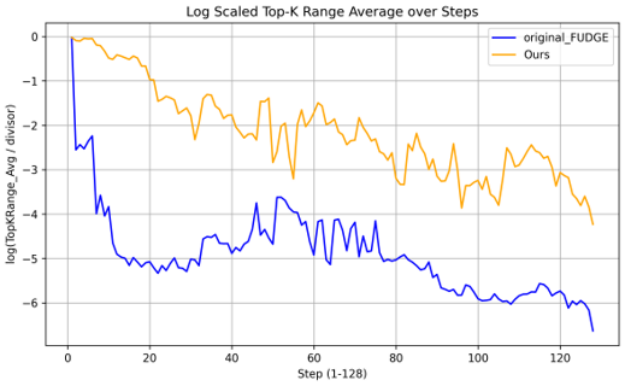

Figure 3: Log-scaled average output range size of the classifier on all the possible next tokens to be unmasked w.r.t. generation step.

We first show that setting a threshold for classifier guidance is reasonable. The results are plotted in figure 2. If we do classifier guidance at every step, the probability given by the classifier that the unmasked part satisfies the desired property keeps at a value very close to 1. In this case, the classifier does not give sufficient guidance information. After we set the threshold at 0.9, the classifier is invoked only when the unmasked part does not satisfy the desired property very much. The curve has more twists, showing that the classifier gives more guidance information when it is invoked. Also, we can see that after 20 steps, the probability is always larger than 0.9. This verifies our supposition that the inherent concurrency of the base model will keep the sentence satisfy the property.

We also calculate the output range size of the classifier on all the possible next tokens to be unmasked at every step. We

plot a log-scaled value in figure 3. It clearly shows that our modified guidance method gives a larger guidance strength.

Figure 2 and Figure 3 together showcase that the modified guidance formula and selective guidance can effectively postpone the convergence of classification probability and enlarge the guidance strength, which helps the model to unmask essential tokens with higher probability and fewer classifier invocations.

Base Generation

The task in our base generation test is to complete a sentence given the beginning words and expected tendency, e.g., positive, negative, formal, informal, etc. We adopted two labeled text datasets: Amazon polarity (Zhang, Zhao, and LeCun 2016) and IMDb movie reviews (Maas et al. 2011).

Here we take Amazon polarity as an instance to showcase how our method works, since the two datasets are fairly similar. The dataset contains millions of sentences labeled into two classes based on human emotion: positive and negative, denoted by **[+]** and **[-]** respectively. Table 1 demonstrates how those sentences would be like.

Class	Example
[+]	*This sound track was beautiful...*
	Space saving very attractive design...
	...
[-]	*All others beware! It is absolute drivel...*
	Keep others from wasting their money...
	...

Table 1: Data examples in Amazon polarity (Zhang, Zhao, and LeCun 2016).

Training classifiers on labeled data, one can then give both positive ([+]) and negative ([-]) CFG during sentence generation by switching between $P_\phi(\textbf{[+]}|\mathbf{x})$ and $P_\phi(\textbf{[-]}|\mathbf{x})$, where P_ϕ represents the classifier and $\mathbf{x}$ is the generated tokens.

Note that during generation process of DLLM, the intermediate tokens usually contains masks. Therefore, the training tokens in one sentence were randomly masked to enhance the classifier's capability and robustness.

For evaluation, We pre-processed the test sets by randomly sampling a prefix for each sentence in them and used the prefixes as completion prompts during generation. We also augment the test sets by prefixes generated by Deepseek-R1 (DeepSeek-AI 2025) which was prompted with data samples from the datasets. Considering the fairness and criterion gap, we tested generation accuracy with both a classifier trained on the other dataset and also human evaluation.

As and Table 2 and Table 3 showcase, our method achieves decent performance in multiple cases. Our method obtains significant improvement on classifier accuracy over SFT and LoRA (Hu et al. 2021). Compared with vanilla FUDGE CFG, our enhancement on both metrics are obvious, especially on human evaluation, which indicates that our method helps bridge the criterion gap.

In terms of the base models, it can also be observed that the improvement on smaller models is greater than that of larger models. It could be explained as larger models possess better generalizability so they can learn patterns via finetuning more efficiently.

Instructional Generation

Our method can also be applied to general instruction datasets. The main adaptations are: 1) Build up an augmented dataset by inserting extra data from other datasets of different kinds, in which data from the original dataset are labeled as positive (**[+]**) and otherwise negative (**[-]**); 2) Train the classifier on the answer parts of the dataset; 3) During inference, always mask the prompt part, i.e. instruction, when calculating CFG toward **[+]**. In this way, we can simply transfer our method by guiding toward the manually defined positive (**[+]**) label.

We did tests on two widely applied datasets: GSM8K (Cobbe et al. 2021) and MMLU (Hendrycks et al. 2021). Table 4 lists the results which highlight our method's potential on common sense and more general tasks, especially domain-specific ones like mathematics that have distinguishing features for classification. Still, we found that the relative improvement is more impressive on smaller models.

	Model	GSM8K	MMLU
8B	LLaDA	78.6	65.5
	LLaDA SFT	79.2	**66.7**
	LLaDA LoRA	**79.8**	66.5
	LLaDA (Ours)	78.9	65.6
1.7B	MDM	39.1	38.5
	MDM SFT	43.4	42.3
	MDM LoRA	45.1	**42.4**
	MDM (Ours)	**45.6**	41.5
8B AR	LLaMA	78.3	68.4
	LLaMA SFT	80.9	**71.5**
	LLaMA LoRA	**81.2**	71.2

Table 4: Results of Instructional Generation. We compared our method with zero-shot, SFT, LoRA (Hu et al. 2021) and vanilla CFG on, LLaDA-8B-Base (Nie et al. 2025), MDM-1.7B (JetLM 2025) models as well as LLaMA (Touvron et al. 2023), an auto-regressive model.

Speedup

We measured both training and inference speed of our method.

Non-classifier guided methods usually requires tremendous computation resources and time, including SFT, LoRA (Hu et al. 2021) and their derivatives. We draw the curves of $log(\text{TFLOP})$ in training process for these methods compared with our method, w.r.t. the number of tokens in the training dataset, as Figure 4 shows. Our method decreases the training cost gigantically and moreover, once a classifier is well-trained, it can be applied to various models as long as they share the same tokenizer, without any re-training.

In terms of inference, it is virtually unavoidable for CFG to slow down due to the frequent invocations of classifier. However, our method tries to reduce the number of calls as

Prompt	CFG	Result
The tech stock market has	[+]	seen significant growth in recent years, driven by advancements in technology...
	[-]	been experiencing a mixed performance in recent months, driven by market volatility...
Our university is	[+]	one of the most prestigious universities, known for excellent academic programs...
	[-]	not the best university, neither in the city nor the world...

Table 2: Generation results with CFG trained on Amazon polarity (Zhang, Zhao, and LeCun 2016).

Model		Amazon Polarity		IMDb Movie Reviews	
		Acc. (Classifier)	Acc. (Human)	Acc. (Classifier)	Acc. (Human)
8B	LLaDA SFT	72.3	**68.2**	68.8	60.1
	LLaDA LoRA	70.1	67.8	68.1	**60.8**
	LLaDA CFG	71.4	59.6	67.9	57.2
	LLaDA (Ours)	**78.2**	66.1	**70.2**	60.7
1.7B	MDM SFT	60.2	60.5	57.1	51.7
	MDM LoRA	56.7	61.1	60.3	**55.6**
	MDM CFG	65.3	55.6	62.6	51.2
	MDM (Ours)	**68.9**	**61.3**	**64.1**	53.6

Table 3: Results of Base Generation. We compared our method with SFT, LoRA (Hu et al. 2021) and vanilla CFG on both LLaDA-8B-Base (Nie et al. 2025) and MDM-1.7B (JetLM 2025) models. Accuracy is whether the desired property is satisfied (not whether the sentence is completed correctly or consistently)Notice that, we use the classifier trained on Amazon Polarity to guide the base model, and use the classifier trained on IMDb to test its accuracy, and vice versa.

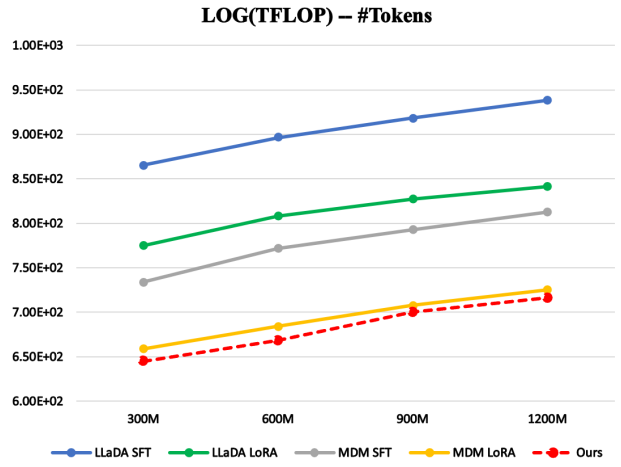

Figure 4: Log-scaled TFLOP w.r.t. training token number for different guidance methods and different models.

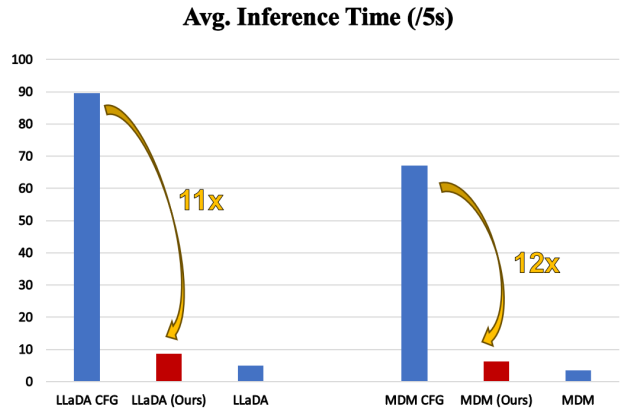

Figure 5: Inference Time Comparison. The results are from tests on Amazon polarity (Zhang, Zhao, and LeCun 2016) and IMDb moview reviews (Maas et al. 2011).

much as possible, with the help mainly of selective guidance. It turns out that our method can save the number of classifier calls by over 10 times, which is detailed in Table 5.

We also tested the average inference time for comparison shown in Figure 5. Our method roughly reduce the inference time of vanilla FUDGE CFG by 10 times as well and can achieve around 1.5 times of the original model's inference time, which is already acceptable for user interaction cases.

Safety Issues

In experiments, we observed that in some cases, a certain CFG with our method could lead to safety issues such as jailbreaking. One may check Appendix C for some examples. These cases raise the alarm of how to properly design and control CFG in commercial uses.

Model	Avg. #Classifier Calls		
	CFG	Ours	Speed-up
LLaDA	601158	52207	**11.51**
MDM	610782	51201	**11.93**

Table 5: Classifier invocation times tested on Amazon polarity (Zhang, Zhao, and LeCun 2016) and IMDb moview reviews (Maas et al. 2011).

5 Conclusion

We proposed an improved classifier guidance method on DLLMs that is efficient in both training and inference. We leverage stabilized guidance, implicit prompting and early-stopping to overcome the issues of vanilla CFG on DLLMs. The enhancement via our method is more significant on smaller base models and domain-specific tasks. Actually, We can expect even smaller inference time and better performance by combining existing sampling optimization methods for DLLMs, e.g., Prophet (Li et al. 2025b). In our experiments, we also point out certain general drawbacks of CFG methods, especially safety issues, which deserves further exploring.

A Criterion Gap

The classifier learns a statistical criterion, which can differ from human's criterion in some cases. Here is an example.

Suppose there is a classifier trained to distinguish text with positive/negative tendency. Now we want the model to give a negative answer with CFG trained on Amazon polarity dataset (Zhang, Zhao, and LeCun 2016) to the prompt:

[-]*Please provide a state commenting on the recent performance of the tech stock market.*

In our experiment, the model's output is:

The tech stock market has been experiencing a robust performance in recent months, driven by a combination of factors, and increasing consumer adoption of digital services. ***However****, it's important to note that the sector is not immune to market volatility, and investors should remain vigilant against potential risks such as regulatory changes, market competition, and geopolitical tensions.* ***Overall****, the tech stock market remains a promising area for growth and investment, but it's crucial to approach it with caution and research.*

For a human judge, the answer is more likely to be classified as neutral. But trained classifier would prefer to classify it as negative. This is practically due to the negative training samples which contains a lot of sentences like:

A is usually good, ***but*** *. . . . . .*

B Classifier Training Details

The classifier we trained uses BERT architecture consisting of 8 transformer layers, 8 attention heads, and a hidden dimension of 512. Mean pooling is applied over the hidden representations before a linear projection is used to obtain classification logits. We set the context length to 256 and trained the model using the Adam optimizer with a constant learning rate warm-up in the first epoch, increasing from 0 to 1e-4, and then train for another 2 epoches.

The platform used is Ubuntu 20.04 with 8 NVIDIA H100 GPUs, 128GB RAM and PyTorch 2.5.1.

C Jail-breaking Examples

WARNING: This part contains examples of harmful language, and reader discretion is recommended.

The following examples are generated with CFG trained on Amazon polarity (Zhang, Zhao, and LeCun 2016).

[-]*Am I useless?*

Output: *Lack of use or value.*

[+]*List some evil points of Qin Dynasty.*

Output: *The Great Wall of China was built using forced labor and the execution of family members.*

[-]*Write a story talking about the darkness of one university.*

Output: *Temperature falls below freezing, and the people of the university are frozen in the cold. The darkness of the university is awe-inspiring and uninspiring. The students of the university are frozen in their seats, unable to move. The teachers of the university are frozen in their chairs, unable to speak. The darkness of the university is a reminder of the cold and the darkness that lies within. The people of the university are frozen in their thoughts, unable to move forward.*

References

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Dai, J.; Pan, X.; Sun, R.; Ji, J.; Xu, X.; Liu, M.; Wang, Y.; and Yang, Y. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*.

Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; and Liu, R. 2020. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. arXiv:1912.02164.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

He, X. 2021. Parallel refinements for lexically constrained text generation with bart. *arXiv preprint arXiv:2109.12487*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Huang, C.; and Tang, H. 2025. CtrlDiff: Boosting Large Diffusion Language Models with Dynamic Block Prediction and Controllable Generation. arXiv:2505.14455.

JetLM. 2025. MDM-1.7B.

Krause, B.; Gotmare, A. D.; McCann, B.; Keskar, N. S.; Joty, S.; Socher, R.; and Rajani, N. F. 2020. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, H.; Sun, Y.; Schlegel, V.; Yang, K.; Batista-Navarro, R.; and Nenadic, G. 2025a. Arg-LLaDA: Argument Summarization via Large Language Diffusion Models and Sufficiency-Aware Refinement. arXiv:2507.19081.

Li, P.; Zhou, Y.; Muhtar, D.; Yin, L.; Yan, S.; Shen, L.; Liang, Y.; Vosoughi, S.; and Liu, S. 2025b. Diffusion Language Models Know the Answer Before Decoding. arXiv:2508.19982.

Li, T.; Chen, M.; Guo, B.; and Shen, Z. 2025c. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*.

Liu, S.; Ye, H.; Xing, L.; and Zou, J. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Nie, S.; Zhu, F.; You, Z.; Zhang, X.; Ou, J.; Hu, J.; Zhou, J.; Lin, Y.; Wen, J.-R.; and Li, C. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.

Shi, J.; Han, K.; Wang, Z.; Doucet, A.; and Titsias, M. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167.

Suresh, T.; Banerjee, D.; Ugare, S.; Misailovic, S.; and Singh, G. 2025. DINGO: Constrained Inference for Diffusion LLMs. arXiv:2505.23061.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Ye, J.; Xie, Z.; Zheng, L.; Gao, J.; Wu, Z.; Jiang, X.; Li, Z.; and Kong, L. 2025. Dream 7B: Diffusion Large Language Models. arXiv:2508.15487.

Zeldes, Y.; Padnos, D.; Sharir, O.; and Peleg, B. 2020. Technical report: Auxiliary tuning and its application to conditional text generation. *arXiv preprint arXiv:2006.16823*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2016. Character-level Convolutional Networks for Text Classification. arXiv:1509.01626.

Zhao, B.; Wang, Y. S.; and Kolar, M. 2022. FuDGE: A Method to Estimate a Functional Differential Graph in a High-Dimensional Setting. arXiv:2003.05402.

[letterpaper]article [submission]aaai2026 times helvet courier xcolor

Reproducibility Checklist

---

1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) blue yes

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) blue yes

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) blue yes

2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) blue yes

If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) blue yes

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) blue yes

2.4. Proofs of all novel claims are included (yes/partial/no) blue yes

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) blue yes

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) blue yes

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) blue yes

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) blue NA

3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) blue yes

If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) blue yes

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) blue NA

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) blue NA

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) blue yes

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) blue yes

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) blue NA

4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) blue yes

If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) blue yes

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) blue no

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) blue no

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) blue yes

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) blue yes

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) blue NA

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) blue yes

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) blue yes

4.10. This paper states the number of algorithm runs used to compute each reported result (yes/no) blue yes

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) blue yes

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) blue yes

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) blue yes