Aligning Compound AI Systems via System-level DPO

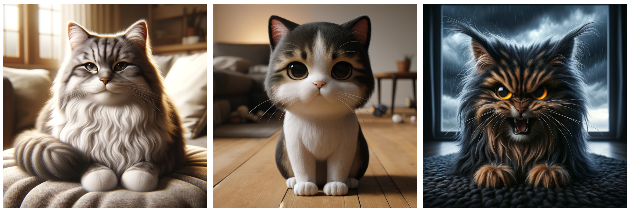
Xiangwen Wang[1,2]*, Yibo Jacky Zhang[1]*, Zhoujie Ding[1], Katherine Tsai[1], Sanmi Koyejo[1]
[1]Stanford University, Stanford, CA, USA
[2]University of Science and Technology of China, Hefei, China
wangxiangwen@mail.ustc.edu.cn, yiboz@stanford.edu, d1ng@stanford.edu, tsaikl@stanford.edu, sanmi@cs.stanford.edu

Abstract

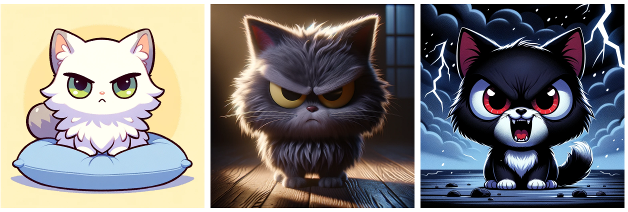
Compound AI systems, comprising multiple interacting components such as LLM agents and external tools, demonstrate state-of-the-art results across diverse tasks. It is hence crucial to align components within the system to produce consistent results that match human expectations. However, conventional alignment methods, such as Direct Preference Optimization (DPO), are not directly applicable to compound AI systems. These challenges include the non-differentiable interactions between components, making end-to-end gradient optimization infeasible. Additionally, system-level preferences cannot be directly translated into component-level preferences, further complicating alignment. We address the issues by formulating compound AI systems as Directed Acyclic Graphs (DAGs), capturing the connections between agents and the data generation processes. We propose a system-level DPO (SysDPO) to jointly align compound systems by adapting the DPO to operate on these DAGs. We study the joint alignment of an LLM and a diffusion model to demonstrate the effectiveness of our approach. Our exploration provides insights into the alignment of compound AI systems and lays a foundation for future advancements.

1 Introduction

Compound AI systems, which consist of multiple interacting AI components[1], serve as promising frameworks to push beyond the model capabilities and achieve state-of-the-art performance (Zaharia et al. 2024; Chen et al. 2024; Kandogan et al. 2024; Lin et al. 2024). For example, ChatGPT integrates a large language model (LLM), a DALL-E image generator, a web browser plugin, and more (Achiam et al. 2023). A multi-agent system consisting of multiple LLMs working collaboratively, e.g., Mixture-of-Agents (MoA), achieves improved performance compared to a single agent (Wang et al. 2024). A Retrieval-Augmented Generation (RAG) system combines large language models with information retrieval capabilities and is able to answer time-sensitive queries. A multi-LLM routing system includes a router that dynamically selects among a diverse set of models to maximize the overall performance (Hu et al. 2024).



(a) Calm Cat (b) Slightly Irritated Cat (c) Very Angry Cat



(d) Slightly Annoyed Cat (e) Angry Cat (f) Furious Cat

Figure 1: This example illustrates the challenges in a compound system composed of GPT-4 and the image generator DALL-E. Given the user prompt to GPT-4, ”Generate three separate images of a cat being progressively angrier.” The first row (a–c) shows the results from the first query, while the second row (d–f) represents the results from another query. The captions under each image summarize the prompts generated by GPT-4 for DALL-E (complete prompts shown in Appendix). The prompts in both versions reflect a progression in anger. Similarly, DALL-E accurately generates the images following the given prompts. However, the first row fails to demonstrate a clear visual progression of anger compared to the second row, highlighting GPT-4’s inconsistent collaboration with DALL-E. Furthermore, our experiments (Section 4) show that Llama-3-8B and Stable Diffusion XL achieve correct results only 32% of the time on similar tasks.

Compound AI systems utilize LLMs as the base models and further integrate other models to complete more sophisticated tasks that exceed the capability of a single LLM.

It is, therefore, crucial to ensure that the outputs of a compound AI system align with human preferences and that each component within the system is aligned to collaborate effectively (Lin et al. 2024). However, such coordination does not come naturally by simply integrating multiple pre-trained models; we demonstrate a failure case of the coordination between an LLM (GPT-4) and a diffusion model (DALL-E) in Figure 1. This demonstrates the critical need to develop a

---

*These authors contributed equally.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] In compound AI systems, *components* include *models* and *agents*, and these terms are used interchangeably in this work.

new framework to align compound AI systems.

While there are many effective ways to align monolithic models with human preference (Rafailov et al. 2024; Ziegler et al. 2019; Bai et al. 2022), aligning compound systems remains an open problem. Standard methods such as Direct Preference Optimization (DPO) (Rafailov et al. 2024) and Reinforcement Learning from Human Feedback (RLHF) (Ziegler et al. 2019; Bai et al. 2022) are not directly applicable to compound systems for three primary reasons. First, components in a compound AI system communicate in a non-differentiable way such as through plain text, which prohibits an end-to-end gradient optimization or RLHF. Second, aligning each component separately is problematic because the overall system's preferences cannot be decomposed into the preferences of individual components. Effective collaboration among components is critical but not easily captured by aligning them individually. Third, while alignment datasets may exist for the system's overall task, they are often not available for the unique sub-tasks of individual components.

In light of these challenges, there is an urgent need to develop methodologies for aligning compound AI systems. While recent studies have explored prompting techniques and instruction tuning approaches (Yuksekgonul et al. 2024; Lin et al. 2024; Shinn et al. 2024), these solutions only partially address the fundamental challenges. To tackle these challenges, we make the following contributions:

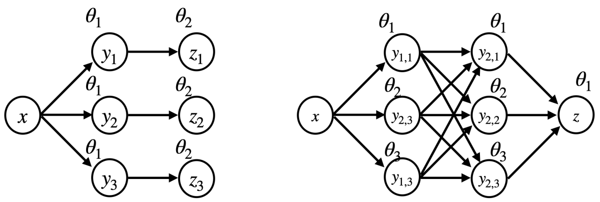
- We formally define the problem of preference learning and alignment of compound AI system and then propose SysDPO to align the entire compound AI system;
- We show how SysDPO can be applied to align a compound AI system composed of an LLM agent and a text-to-image diffusion model;
- We demonstrate that aligning compound AI systems increases the success rate in handling complex instructions and coordinating components.

These results deepen our understanding of the alignment challenges in compound AI systems and provide a foundation for future research.

2 The SysDPO Framework

In this section, we introduce the SysDPO pipeline. We start by modeling the structure of compound AI systems as Directed Acyclic Graphs (DAGs), which encode both the connections between agents and the flow of the underlying data generation process. The DAG structure enables us to factorize the joint probability of generated outputs into several components, resolving the non-differentiability issue when aligning multiple agents. We then define a DPO-based loss function that can be optimized from end-to-end simply via gradient descent. The end-to-end optimization ensures that each agent is aligned with user-defined preferences. Below, we outline the key steps in the pipeline:

1. System Representation. We represent the compound AI system as a Directed Acyclic Graph (DAG).We define nodes as $x, \{y_i\}_{i\in I}, \{z_j\}_{j\in J}$, where x is the input, y_i for $i \in I$ are intermediate outputs and z_j for $j \in J$ are final outputs. Except for the input x, each node represents a generated output, given by a single model or an external tool based on some other nodes. We define the set of all generated outputs as $s = \{y_i, z_j\}_{i\in I, j\in J}$. The directed edges represent the flow of the generated data between components.



(a) Example 1 (b) Example 2

Figure 2: **Example 1.** (LLM+Diffusion Models) The user gives a prompt x which is processed by the LLM θ_1 to produce three captions y_1, y_2, y_3 for the diffusion model. The diffusion model θ_2 is called three times to generate images z_1, z_2, z_3 based on three captions. **Example 2.** (Mixture-of-agents) The two-layered MoA composed of three models $\theta_1, \theta_2, \theta_3$. The instruction x is sent to the three models in the first layer, and three text outputs $y_{1,1}, y_{1,2}, y_{1,3}$ are generated respectively. Then, all of the previous texts $(x, y_{1,1}, y_{1,2}, y_{1,3})$ are sent to the second layer, and three outputs $y_{2,1}, y_{2,2}, y_{2,3}$ are generated respectively. We omit the arrow from x to the later layer for better readability. Finally, the output from the second layer along with the input, i.e., $(x, y_{2,1}, y_{2,2}, y_{2,3})$ is sent to the final model to obtain the final output.

Let us consider two concrete examples of a compound AI system. The first example involves an LLM and a diffusion model as shown in Figure 1 with the user prompt x being "generate three separate images of ...." The DAG of this example is shown in Figure 2 (a). The second example is MoA (Wang et al. 2024). It leverages the collective power of multiple LLMs through a layered architecture, where each agent combines outputs from the preceding layer as the auxiliary information to generate responses. We formulate the DAG for a two-layered MoA composed of three models, as shown in Figure 2 (b).

2. Probability Factorization. The DAG structure encodes the conditional independence of the generated data (Pearl 2009), enabling the decomposition of the probability of the generated data into multiple terms:

$$p_\theta(s|x) \quad = \prod_{i\in I, j\in J} p_{\theta_i}(y_i|\mathtt{P}(y_i)) \ \cdot \ p_{\theta_j}(z_j|\mathtt{P}(z_j)), \quad (1)$$

where $\mathtt{P}(\cdot)$ returns the parent nodes (may include the input x) of a given node in the graph, and $\theta = \{\theta_k : k \in I \cup J\}$ denotes the parameter set of models in the compound AI system. This decomposition, derived from the DAG structure, breaks down the likelihood of system generation into a product of multiple terms, where each term contains a single model, allowing model-dependent optimization. Take the case of Figure 2 (a) as an example and denoting the set of generated contents as $s = \{y_1, y_2, y_3, z_1, z_2, z_3\}$, we have $p(s|x) = \prod_{i=1}^{3} p_{\theta_1}(y_i|x) \cdot p_{\theta_2}(z_i|y_i)$.

For external tool θ_i integrated within the system, the probability factorization $p_{\theta_i}(y_i|\mathtt{P}(y_i))$ is set to 1, assuming that external tools provide deterministic outputs.

3. Preference Dataset Construction. SysDPO optimizes for pairwise preferences by leveraging a preference dataset. The dataset can be obtained in the following way: given a query x, the system generates two versions of the responses, which include outputs of every agent. We label the preferred set as s^w, and the not-preferred set as s^l.

4. Loss Function Design. Given such a dataset D composed of preference pairs (x, s^w, s^l) and a compound AI system formulated as a DAG, we can apply DPO to align the system (Rafailov et al. 2024):

$$L(\theta) = \\ -\mathbb{E}_{(x,s^w,s^l)\sim D}\left[\log\sigma\left(\beta\log\frac{p_\theta(s^w|x)}{p_{\bar\theta}(s^w|x)} - \beta\log\frac{p_\theta(s^l|x)}{p_{\bar\theta}(s^l|x)}\right)\right], \tag{2}$$

where $\bar\theta$ denotes the collection of reference models, $\sigma(\cdot)$ stands for the sigmoid function. By decomposing p_θ via (1) in the DPO loss, we derive a differentiable loss function tailored for compound AI systems, which we refer to as the SysDPO loss. Unlike the original DPO loss, which optimizes individual models, SysDPO integrates probability decomposition to capture interactions between multiple components in compound AI systems.

3 Application: Compound AI System of a LLM and a Diffusion Model

In this section, we apply SysDPO to a group-image-generation application with an example in Figure 1, which involves an LLM ψ and a Diffusion Model ϕ. For a single input x provided to the system, the LLM generates an intermediate output y, which can be parsed to multiple captions $y_1, y_2, \ldots, y_n$. Each $y_i,\ i = 1, \ldots, n$ serves as a prompt for the diffusion model. The diffusion model is then queried n times, generating images $z_1, z_2, \ldots, z_n$ as the final outputs. This multi-step process is modeled as a DAG whose special case $(n = 3)$ is shown in Figure 2 (a), and it allows us to decompose the generation process by

$$p(s|x) = p_\psi(y|x)\cdot\prod_{i=1}^{n} p_\phi(z_i|y_i). \tag{3}$$

Note that, for better readability, we adopt a different notation for the models in this section as opposed to the notation used in Section 2. Apply the decomposition of probability equation 3 to the loss function equation 2, we get the joint loss function of this system

$$L(\psi,\phi) = -\mathbb{E}_{(x,s^w,s^l)\sim D} \\ \left[\log\sigma\left(\beta\left(\log\frac{p_\psi(y^w|x)}{p_{\bar\psi}(y^w|x)} + \sum_i^n \log\frac{p_\phi(z_i^w|y_i^w)}{p_{\bar\phi}(z_i^w|y_i^w)}\right)\right.\right. \\ \left.\left. - \beta\left(\log\frac{p_\psi(y^l|x)}{p_{\bar\psi}(y^l|x)} + \sum_i^n \log\frac{p_\phi(z_i^l|y_i^l)}{p_{\bar\phi}(z_i^l|y_i^l)}\right)\right)\right], \tag{4}$$

where $s^w = \{y^w, z_1^w, z_2^w, \ldots z_n^w\}$, and $\bar\psi, \bar\phi$ are reference models. The language model's generation likelihood $p_\psi(y|x)$ is accessible, while the diffusion model's $p_\phi(z|y)$ is not. The following subsection will handle this challenge by delving into the generation process of diffusion models.

3.1 Handling the Diffusion Model

To obtain the diffusion model's generation likelihood, we build upon (Wallace et al. 2024), which applies DPO to denoising diffusion probabilistic models (Ho, Jain, and Abbeel 2020), and extend it to accommodate our framework. Details of the derivation and the theorem are in Appendix A.

A diffusion model learns to reverse a diffusion process, represented by a sequence $z_{0:T} := (z_0, z_1, \ldots, z_T)$, where the original image z_0 is gradually transformed into standard Gaussian noise z_T over T steps. By learning to reverse this process, the model generates images by progressively denoising z_T, starting from noise and reconstructing the original image z_0. The likelihood of the reverse process is

$$p_\phi(z_{0:T}|y) = p(z_T)\prod_{t=1}^{T} p_\phi(z_{t-1}|z_t, y),$$

where each $p_\phi(z_{t-1}|z_t, y)$ is a Gaussian density function.

However, the diffusion model does not directly provide the likelihood p_ϕ, even for a small Gaussian step $p_\phi(z_{t-1}|z_t, y)$. To this end, Ho, Jain, and Abbeel (2020) proposed a denoiser ϵ_ϕ, which predicts the original image from a noisy input and can be applied to approximate the likelihood. Such denoiser can be learned from data by optimizing the following objective function:

$$\ell_\epsilon(\phi; t, z_{i,t}^w, y_i^w) := \left[w_t\left\|\epsilon - \epsilon_\phi(z_{i,t}^w, t, y_i^w)\right\|^2\right],$$

where w_t is a weight parameter, $z_{i,t}^w$ is the i-th output at timestep t, and ϵ corresponds to the noise added to $z_{i,0}$ from which $z_{i,t}^w$ is derived. Similarly, we use $\ell_\epsilon(\phi; t, z_{i,t}^l, y_i^l)$ to denote the denoising loss for the losing data.

We prove the following theorem, which converts equation 4 into a loss function that directly utilizes the denoiser loss function, thereby making the loss function optimizable.

Theorem 1. *The loss function (4) is upper bounded by*

$$L(\psi,\phi) \le -\mathbb{E}_{(x,s^w,s^l)}\mathbb{E}_t\,\mathbb{E}_{z_{i,t}^w, z_{i,t}^l}\left[\log\sigma\left(\beta\left(\right.\right.\right. \\ \left(\log\frac{p_\psi(y^w|x)}{p_{\bar\psi}(y^w|x)} + T\sum_i\left(-\ell_\epsilon(\phi; t, z_{i,t}^w, y_i^w) + \ell_\epsilon(\bar\phi; t, z_{i,t}^w, y_i^w)\right)\right) - \\ \left.\left.\left.\left(\log\frac{p_\psi(y^l|x)}{p_{\bar\psi}(y^l|x)} + T\sum_i\left(-\ell_\epsilon(\phi; t, z_{i,t}^l, y_i^l) + \ell_\epsilon(\bar\phi; t, z_{i,t}^l, y_i^l)\right)\right)\right)\right)\right].$$

Thus, we obtain a tractable loss function for SysDPO.

4 Experiments

We evaluate the effectiveness of SysDPO alignment in a compound AI system described in section 3. We train and evaluate the system on a dataset of multi-modal progression tasks, where the system generates sequences of images with a specific scene-related attribute that varies progressively. Examples of inputs and outputs are provided in Appendix E. Our evaluation focuses on the coherence among images and their alignment with holistic preferences.

Dataset Construction. We constructed a custom dataset using the following steps:

1. **Attribute Selection:** We use a regressor from Zhuang, Koyejo, and Schwing (2021) which gives scores from $[0, 1]$ to images based on 40 distinct scene-related attributes (e.g., brightness, coldness, fog density).

2. **Instruction Design:** For each attribute, we query GPT-4 to generate 250 user prompts of generating a sequence of images representing the progression of the intensity of that attribute. To ensure the diversity of user prompts, we generate prompts using four distinct prompt styles from Qin et al. (2024). Details are provided in Appendix D.
3. **Constructing Chosen and Rejected Pairs:** For each user prompt, four image sequences are generated and ranked using the Preference Score q, described below in equation 5. Six comparison pairs are constructed from the four samples. The instance among the two pairs with the higher preference score is marked as the preferred. The dataset contains a total of 6000 comparison pairs.

Preference Score. To compare the generated image sequences, we define a *preference score* q that evaluates both order consistency and distribution evenness. This metric is based on the attribute scores assigned to the images by the regressor from Zhuang, Koyejo, and Schwing (2021). Given a sequence of three images with attribute scores a_1, a_2, and a_3, the Preference Score q is computed as:

$$q = -\left(a_1 - a_3 + |a_2 - (a_1 + a_3)/2|\right) \tag{5}$$

Sequences with higher q values are preferred, as they reflect correct ordering and smoother distributions. Conversely, reversed or uneven sequences result in lower q.

For further details, including examples illustrating the calculation of q, please refer to Appendix C.

Models. For dataset construction and evaluation, we use an instruction-tuned Llama-3-8B model (AI@Meta 2024) as the language model. To generate image sequences for constructing chosen and rejected samples in the dataset, we employ Stable Diffusion XL (SDXL) (Podell et al. 2023). For training purposes, we use Stable Diffusion 1.5 (Rombach et al. 2022) which provides a balance between computational efficiency and generation quality.

Evaluation. The performance of the system is evaluated using two metrics. The first metric is the **Average Preference Score** across all generated sequences from the test dataset. The second evaluation metric is the **Order Consistency Ratio**, measuring the proportion of generated sequences in the correct order, i.e., where $a_1 < a_2 < a_3$.

Baselines. To evaluate the effectiveness of the proposed SysDPO joint alignment approach, we compare it against four baseline methods.

1. **System Before Alignment.** The first baseline represents the system prior to applying SysDPO. Notably, Llama-3-8B-it is instruction-tuned, so that it serves as a baseline for conventional separately aligned systems.
2. **Best-of-4 Sampling Baseline.** For this baseline, we sample four image sequences generated by the system without optimization. For each user prompt, we select the best-performing sequence based on the Preference Score. The average of the selected sequences is reported.
3. **Only Train Language Model or Diffusion Model.** In this baseline, we freeze the weights of the diffusion model or language model and train only another model using the dataset and proposed loss function of SysDPO.

Results. This section presents the performance of the proposed SysDPO compared to the baselines. We evaluate the system using the Preference Score and Order Consistency Ratio. Examples of system outputs before and after training can be found in Appendix E.[2]

Table 1: Performance comparison of the proposed method and baselines. Higher Preference Scores (Pref. Score) and higher Order Consistency Ratios (OC Ratio) are better.

Method	**Pref. Score**	**OC Ratio**
SysDPO (Proposed)	0.25	70%
System Before Alignment	-0.20	32%
Best-of-Sampling	0.16	67%
Only Train Language Model	0.23	65%
Only Train Diffusion Model	-0.03	35%

The results in Table 1 demonstrate the importance of alignment in compound AI systems and the effectiveness of the proposed SysDPO alignment approach. The "System Before Alignment" baseline achieves poor performance, with a low Preference Score and a low Order Consistency Ratio (32%), indicating that conventionally instruction-tuned components are insufficient for ensuring coherent collaboration in compound systems. The "Only Train Language Model" baseline achieves significantly better results than the "Only Train diffusion Model", with a Preference Score of 0.23 and a Ratio of 65%. This is due to the LLM's role in generating captions that control output sequences, influencing the overall progression and coherence of the system. SysDPO achieves the best Preference Score (0.25) and the highest Order Consistency Ratio (70%). These results validate the effectiveness of our SysDPO algorithm, demonstrating its ability to optimize both components together for superior performance in generating coherent image sequences.

5 Discussion and Future Work

Our preliminary investigations indicate that the proposed formulation and methodology are promising for aligning compound AI systems. However, further experimental investigations are necessary to evaluate its potential comprehensively. For instance, how does our approach compare to existing techniques, such as instruction tuning and prompting strategies? Additionally, the scalability of our method to more complex applications, where the number of components and interactions grows significantly, remains open.

Despite these open questions, our work establishes a solid foundation for aligning compound AI systems as cohesive entities. We believe that the insights and framework presented here pave the way for promising advancements in this area of research.

---

[2] We use LoRA to train the language model, reducing memory overhead. The experiments were performed on 2 A100 GPUs, requiring around 4 hours of training.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback and suggestions. SK acknowledges support from NSF grants 2046795 and 2205329, IES grant R305C240046, the MacArthur Foundation, Stanford HAI, OpenAI, and Google. KT acknowledges support from the NSF Graduate Research Fellowship.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 Model Card.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Chen, L.; Davis, J. Q.; Hanin, B.; Bailis, P.; Stoica, I.; Zaharia, M.; and Zou, J. 2024. Are More LLM Calls All You Need? Towards the Scaling Properties of Compound AI Systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Hu, Q. J.; Bieker, J.; Li, X.; Jiang, N.; Keigwin, B.; Ranganath, G.; Keutzer, K.; and Upadhyay, S. K. 2024. ROUTERBENCH: A Benchmark for Multi-LLM Routing System. *arXiv preprint arXiv:2403.12031*.

Kandogan, E.; Rahman, S.; Bhutani, N.; Zhang, D.; Chen, R. L.; Mitra, K.; Gurajada, S.; Pezeshkpour, P.; Iso, H.; Feng, Y.; et al. 2024. A Blueprint Architecture of Compound AI Systems for Enterprise. *arXiv preprint arXiv:2406.00584*.

Lin, M.; Sheng, J.; Zhao, A.; Wang, S.; Yue, Y.; Wu, Y.; Liu, H.; Liu, J.; Huang, G.; and Liu, Y.-J. 2024. LLM-based Optimization of Compound AI Systems: A Survey. *arXiv preprint arXiv:2410.16392*.

Pearl, J. 2009. *Causality*. Cambridge university press.

Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Qin, J.; Wu, J.; Chen, W.; Ren, Y.; Li, H.; Wu, H.; Xiao, X.; Wang, R.; and Wen, S. 2024. Diffusiongpt: LLM-driven text-to-image generation system. *arXiv preprint arXiv:2401.10061*.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10684–10695.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Wallace, B.; Dang, M.; Rafailov, R.; Zhou, L.; Lou, A.; Purushwalkam, S.; Ermon, S.; Xiong, C.; Joty, S.; and Naik, N. 2024. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8228–8238.

Wang, J.; Wang, J.; Athiwaratkun, B.; Zhang, C.; and Zou, J. 2024. Mixture-of-Agents Enhances Large Language Model Capabilities. *arXiv preprint arXiv:2406.04692*.

Yuksekgonul, M.; Bianchi, F.; Boen, J.; Liu, S.; Huang, Z.; Guestrin, C.; and Zou, J. 2024. TextGrad: Automatic” Differentiation” via Text. *arXiv preprint arXiv:2406.07496*.

Zaharia, M.; Khattab, O.; Chen, L.; Davis, J. Q.; Miller, H.; Potts, C.; Zou, J.; Carbin, M.; Frankle, J.; Rao, N.; et al. 2024. The shift from models to compound ai systems. *Berkeley Artificial Intelligence Research Lab. Available online at: https://bair. berkeley. edu/blog/2024/02/18/compound-ai-systems/(accessed February 27, 2024)*.

Zhuang, P.; Koyejo, O.; and Schwing, A. G. 2021. Enjoy your editing: Controllable GANs for image editing via latent space navigation. *arXiv preprint arXiv:2102.01187*.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Appendix

A Details of Diffusion Models and Proof of Theorem 1

In this section of the appendix, we provide a detailed explanation of the DDPM diffusion model and the derivation of Theorem 1.

A.1 Denoising Diffusion Probabilistic Model (DDPM)

DDPM (Ho, Jain, and Abbeel 2020) is a widely used class of diffusion model. Below is a highlight of the key ingredient we need for DPO for DDPM (Wallace et al. 2024), with our framework.

Given a real image z_0, consider a diffusion process, which we call the forward process, gradually making the original image into Gaussian noise z_T after T steps, i.e.,

$$z_0 \to z_1 \to z_2 \to \cdots \to z_T \sim \mathcal{N}(0, I).$$

The goal of the diffusion model ϕ is to reverse this process that recovers an image from noise. The forward process and the reverse process are denoted respectively as

$$q(z_{0:T}|y), \qquad p_\phi(z_{0:T}|y),$$

where y is the context, i.e., the prompt to the diffusion model.

Note that both the forward and backward processes are Markovian, and in particular we have the nice property that the forward process

$$q(z_{0:T}|y) = q(z_0|y) \prod_{t=1}^{T} q(z_t|z_{t-1}), \qquad \text{where each } q(z_t|z_{t-1}) \text{ is a Gaussian.}$$

Similarly, the reverse process

$$p_\phi(z_{0:T}|y) = p(z_T) \prod_{t=1}^{T} p_\phi(z_{t-1}|z_t, y), \qquad \text{where each } p_\phi(z_{t-1}|z_t, y) \text{ is a Gaussian.} \tag{6}$$

In this formulation, the ideal goal for the diffusion model is that $q(z_{0:T}|y) = p_\phi(z_{0:T}|y)$. However, this is not easy to optimize directly. With some analysis, the DDPM paper (Ho, Jain, and Abbeel 2020) proposes to minimize for

$$D_{KL}(q(z_{t-1}|z_t, z_0, y) \| p_\phi(z_{t-1}|z_t, y)) \quad \text{for} \quad t \sim \mathcal{U}([T]), z_0 \sim q(z_0|y),$$

where $\mathcal{U}(\cdot)$ denotes the uniform distribution on a set, and $[T]$ denotes the set of $\{1, 2, \ldots, T\}$. This is done by learning a denoiser ϵ_ϕ operating in the following way. For a real image $z_0 \sim q(z_0|y)$, we sample noise $\epsilon \sim \mathcal{N}(0, I)$, and have

$$z_t(z_0, \epsilon) = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \tag{7}$$

where $\bar{\alpha}_t$ is some parameter such that $z_t \sim q(z_t|z_0)$. Then, the denoiser predicts the noise ϵ that is added to the z_0. I.e.,

$$\epsilon_\phi(z_t(z_0, \epsilon), t, y) \text{ aims to predict } \epsilon.$$

The denoiser ϵ_ϕ is essentially a reparameterization of the mean of $p_\phi(z_{t-1}|z_t, y)$.

The **key ingredient** is that, as shown in (Ho, Jain, and Abbeel 2020),

$$D_{KL}(q(z_{t-1}|z_t, z_0, y) \| p_\phi(z_{t-1}|z_t, y)) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} \left[w_t \left\| \epsilon - \epsilon_\phi(z_t(z_0, \epsilon), t, y) \right\|^2 \right] + C, \tag{8}$$

where w_t is a weight parameter and C is a constant independent of model ϕ.

Therefore, modeling ϵ_ϕ by a neural net, the DDPM model ϕ is trained to minimize the above objective averaged over samples of y, z_0, ϵ, t.

A.2 Dealing with the Diffusion Model in SysDPO

Recall in the main text we obtain the System DPO loss function as equation 4:

$$L(\psi, \phi) = -\mathbb{E}_{(x, s^w, s^l) \sim D} \left[\log \sigma \left(\beta \left(\log \frac{p_\psi(y^w|x)}{p_{\bar{\psi}}(y^w|x)} + \sum_i^n \log \frac{p_\phi(z_i^w|y_i^w)}{p_{\bar{\phi}}(z_i^w|y_i^w)} \right) - \beta \left(\log \frac{p_\psi(y^l|x)}{p_{\bar{\psi}}(y^l|x)} + \sum_i^n \log \frac{p_\phi(z_i^l|y_i^l)}{p_{\bar{\phi}}(z_i^l|y_i^l)} \right) \right) \right].$$

The next step is to convert the likelihood of the diffusion model p_ϕ to something optimizable.

The first step is to consider the generated image as the whole process, i.e.,

$$z_{i,0:T} := \{z_{i,0}, z_{i,1}, \ldots, z_{i,T}\},$$

where $z_{i,0}$ is the generated image, while the others are things in the middle. Following the same notation, we denote

$$z_{i,t-1,t} := \{z_{i,t-1}, z_{i,t}\}.$$

The preference is considered to be given to every process that generates z_0 as the end outcome. Following (Wallace et al. 2024), we have

$$\begin{aligned}L(\theta,\phi) = -\mathbb{E}_{(x,s^w,s^l)\sim D}\Bigg[\log\sigma\Bigg(&\beta\mathbb{E}_{z^w_{i,1:T}\sim q(z^w_{i,1:T}|z^w_{i,0}), z^l_{i,1:T}\sim q(z^l_{i,1:T}|z^l_{i,0})}\\ &\Bigg(\left(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + \sum_i \log\frac{p_\phi(z^w_{i,0:T}|y^w_i}{p_{\bar\phi}(z^w_{i,0:T}|y^w_i)}\right) - \left(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + \sum_i \log\frac{p_\phi(z^l_{i,0:T}|y^l_i)}{p_{\bar\phi}(z^l_{i,0:T}|y^l_i)}\right)\Bigg)\Bigg)\Bigg].\end{aligned}$$

Recall the factorization of the reverse process (equation 6), we have

$$\begin{aligned}L(\theta,\phi) = -\mathbb{E}_{(x,s^w,s^l)\sim D}\Bigg[\log\sigma\Bigg(&\beta\mathbb{E}_{z^w_{i,1:T}\sim q(z^w_{i,1:T}|z^w_{i,0}), z^l_{i,1:T}\sim q(z^l_{i,1:T}|z^l_{i,0})}\\ &\Bigg(\left(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + \sum_i\sum_{t=1}^T \log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}\right) - \left(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + \sum_i\sum_{t=1}^T \log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}\right)\Bigg)\Bigg)\Bigg].\end{aligned}$$

Note that $\sum_{t=1}^T = T\mathbb{E}_{t\sim\mathcal{U}([T])}$ for t is a random variable uniformly distributed on $1, 2, \ldots, T$. Simply denoting $\mathbb{E}_{t\sim\mathcal{U}([T])}$ as E_t, we have

$$\begin{aligned}L(\theta,\phi) &= -\mathbb{E}_{(x,s^w,s^l)\sim D}\Bigg[\log\sigma\Bigg(\beta\mathbb{E}_{z^w_{i,1:T}\sim q(z^w_{i,1:T}|z^w_{i,0}), z^l_{i,1:T}\sim q(z^l_{i,1:T}|z^l_{i,0})}\\ &\Bigg(\left(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i \mathbb{E}_t\log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}\right) - \left(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + T\sum_i \mathbb{E}_t\log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}\right)\Bigg)\Bigg)\Bigg]\\ &= -\mathbb{E}_{(x,s^w,s^l)\sim D}\Bigg[\log\sigma\Bigg(\beta\mathbb{E}_{z^w_{i,1:T}\sim q(z^w_{i,1:T}|z^w_{i,0}), z^l_{i,1:T}\sim q(z^l_{i,1:T}|z^l_{i,0})}\ \mathbb{E}_t\\ &\Bigg(\left(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i \log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}\right) - \left(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + T\sum_i \log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}\right)\Bigg)\Bigg)\Bigg].\end{aligned}$$

Next, we may further simplify the equation by switching $\mathbb{E}_{z^w_{i,1:T}\sim q(z^w_{i,1:T}|z^w_{i,0}), z^l_{i,1:T}\sim q(z^l_{i,1:T}|z^l_{i,0})}$ and $\mathbb{E}_t$ in the above, i.e.,

$$\begin{aligned}L(\theta,\phi) &= -\mathbb{E}_{(x,s^w,s^l)\sim D}\Bigg[\log\sigma\Bigg(\beta\mathbb{E}_t\ \mathbb{E}_{z^w_{i,1:T}\sim q(z^w_{i,1:T}|z^w_{i,0}), z^l_{i,1:T}\sim q(z^l_{i,1:T}|z^l_{i,0})}\\ &\Bigg(\left(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i \log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}\right) - \left(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + T\sum_i \log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}\right)\Bigg)\Bigg)\Bigg]\\ &= -\mathbb{E}_{(x,s^w,s^l)\sim D}\Bigg[\log\sigma\Bigg(\beta\mathbb{E}_t\ \mathbb{E}_{z^w_{i,t-1,t}\sim q(z^w_{i,t-1,t}|z^w_{i,0}), z^l_{i,t-1,t}\sim q(z^l_{i,t-1,t}|z^l_{i,0})}\\ &\Bigg(\left(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i \log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t}, y^w_i)}\right) - \left(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + T\sum_i \log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t}, y^l_i)}\right)\Bigg)\Bigg)\Bigg]\end{aligned}$$

The rationale for the above can be illustrated as follows. Consider a random variables $Z_1, \ldots, Z_T$, and any function $f : (Z_{t-1}, Z_t) \to \mathbb{R}$ for any $t \in [T]$. Then, denoting δ^t_s as the indicator function, i.e., $\delta^t_s = 1$ only if $s = t$, and $\delta^t_s = 0$ otherwise, we can derive

$$\begin{aligned}
\mathbb{E}_{Z_{1:T}}\ \mathbb{E}_{t\sim\mathcal{U}([T])}\ f(Z_{t-1},Z_t) &= \mathbb{E}_{Z_{1:T}}\ \mathbb{E}_{t\sim\mathcal{U}([T])}\ \sum_{s=1}^{T}\delta_s^t\cdot f(Z_{s-1},Z_s)\\
&= \mathbb{E}_{t\sim\mathcal{U}([T])}\ \sum_{s=1}^{T}\delta_s^t\cdot \mathbb{E}_{Z_{1:T}} f(Z_{s-1},Z_s)\\
&= \mathbb{E}_{t\sim\mathcal{U}([T])}\ \sum_{s=1}^{T}\delta_s^t\cdot \mathbb{E}_{Z_{s-1},Z_s} f(Z_{s-1},Z_s)\\
&= \mathbb{E}_{t\sim\mathcal{U}([T])}\ \cdot \mathbb{E}_{Z_{t-1},Z_t} f(Z_{t-1},Z_t).
\end{aligned}$$

Next, noting that $q(z^w_{i,t-1,t}|z^w_{i,0}) = q(z^w_{i,t}|z^w_{i,0})\cdot q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})$ (similarly for $q(z^l_{i,t-1,t}|z^l_{i,0})$), we can first sample $z^w_{i,t}$ and then $z^w_{i,t-1}$ separately, i.e.,

$$\begin{aligned}
L(\theta,\phi) = -\mathbb{E}_{(x,s^w,s^l)\sim D}&\Bigg[\log\sigma\Bigg(\beta\mathbb{E}_t\ \mathbb{E}_{z^w_{i,t}\sim q(z^w_{i,t}|z^w_{i,0}),z^l_{i,t}\sim q(z^l_{i,t}|z^l_{i,0})}\ \mathbb{E}_{z^w_{i,t-1}\sim q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t}),z^l_{i,t-1}\sim q(z^l_{i,t-1}|z^l_{i,0},z^l_{i,t})}\\
&\Bigg(\Bigg(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i \log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t},y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t},y^w_i)}\Bigg) - \Bigg(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + T\sum_i \log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t},y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t},y^l_i)}\Bigg)\Bigg)\Bigg)\Bigg].
\end{aligned}$$

Since $-\log\sigma$ is convex, by Jensen's inequality, we have

$$\begin{aligned}
L(\theta,\phi) \le -\mathbb{E}_{(x,s^w,s^l)\sim D}\mathbb{E}_t\ \mathbb{E}_{z^w_{i,t}\sim q(z^w_{i,t}|z^w_{i,0}),z^l_{i,t}\sim q(z^l_{i,t}|z^l_{i,0})}&\Bigg[\log\sigma\Bigg(\beta\mathbb{E}_{z^w_{i,t-1}\sim q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t}),z^l_{i,t-1}\sim q(z^l_{i,t-1}|z^l_{i,0},z^l_{i,t})}\\
\Bigg(\Bigg(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i \log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t},y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t},y^w_i)}\Bigg) - \Bigg(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} &+ T\sum_i \log\frac{p_\phi(z^l_{i,t-1}|z^l_{i,t},y^l_i)}{p_{\bar\phi}(z^l_{i,t-1}|z^l_{i,t},y^l_i)}\Bigg)\Bigg)\Bigg)\Bigg]. \qquad (9)
\end{aligned}$$

Recall that what we have done so far is all for making the diffusion model's log probability efficiently computable. To complete the derivation, it remains to convert the log-probabilities to the denoising loss via equation 8. Specifically, with C being the constant appears in equation 8, we can see that

$$\begin{aligned}
&\mathbb{E}_{z^w_{i,t-1}\sim q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})}\log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t},y^w_i)}{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t},y^w_i)}\\
&= \mathbb{E}_{z^w_{i,t-1}\sim q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})}\left(\log\frac{p_\phi(z^w_{i,t-1}|z^w_{i,t},y^w_i)}{q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})} - \log\frac{p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t},y^w_i)}{q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})}\right)\\
&= -D_{KL}(q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})\|p_\phi(z^w_{i,t-1}|z^w_{i,t},y^w_i)) + D_{KL}(q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})\|p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t},y^w_i))\\
&= -D_{KL}(q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})\|p_\phi(z^w_{i,t-1}|z^w_{i,t},y^w_i)) + C + D_{KL}(q(z^w_{i,t-1}|z^w_{i,0},z^w_{i,t})\|p_{\bar\phi}(z^w_{i,t-1}|z^w_{i,t},y^w_i)) - C\\
&= -\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[w_t\left\|\epsilon-\epsilon_\phi(z_t(z^w_{i,0},\epsilon),t,y^w_i)\right\|^2\right] + \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[w_t\left\|\epsilon-\epsilon_{\bar\phi}(z_t(z^w_{i,0},\epsilon),t,y^w_i)\right\|^2\right].
\end{aligned}$$

To simplify the notation, we denote

$$\ell_\epsilon(\phi;t,z^w_{i,t},y^w_i) := \left[w_t\left\|\epsilon-\epsilon_\phi(z^w_{i,t},t,y^w_i)\right\|^2\right],$$

where the ϵ corresponds to the noise from which $z^w_{i,t}$ is derived (see equation 7). Similarly, we use $\ell_\epsilon(\phi;t,z^l_{i,t},y^l_i)$ to denote the denoising loss for the losing data.

Thus we can write equation 9 as

$$\begin{aligned}
&L(\theta,\phi) \le -\mathbb{E}_{(x,s^w,s^l)\sim D}\mathbb{E}_t\ \mathbb{E}_{z^w_{i,t}\sim q(z^w_{i,t}|z^w_{i,0}),z^l_{i,t}\sim q(z^l_{i,t}|z^l_{i,0})}\\
&\Bigg[\Bigg[\log\sigma\Bigg(\beta\Bigg(\Bigg(\log\frac{p_\theta(y^w|x)}{p_{\bar\theta}(y^w|x)} + T\sum_i(-\ell_\epsilon(\phi;t,z^w_{i,t},y^w_i) + \ell_\epsilon(\bar\phi;t,z^w_{i,t},y^w_i))\Bigg)\\
&\quad - \Bigg(\log\frac{p_\theta(y^l|x)}{p_{\bar\theta}(y^l|x)} + T\sum_i(-\ell_\epsilon(\phi;t,z^l_{i,t},y^l_i) + \ell_\epsilon(\bar\phi;t,z^l_{i,t},y^l_i))\Bigg)\Bigg)\Bigg)\mid t\Bigg]\Bigg].
\end{aligned}$$

Thus, we complete the proof of Theorem 1.

B Complete Prompts in Figure 1

The complete prompts generated by GPT-4 for DALL-E to generate the six images in Figure 1 are as follows.

(a) A realistic image of a calm cat sitting comfortably on a soft cushion. The cat has a serene expression, with its eyes half-closed in contentment. It has fluffy fur, predominantly grey with white patches, and its tail is neatly curled around its body. The background is a cozy room with a hint of sunlight filtering through a window, casting a warm glow on the cat.

(b) A realistic image of a slightly irritated cat sitting on a wooden floor. The cat has a slight frown and its ears are tilted back slightly, indicating mild annoyance. It has short fur, a mix of black and white, and is staring directly at the viewer with wide eyes. The background is simple, with a few scattered cat toys and a small plant, adding to the homey atmosphere.

(c) A realistic image of a very angry cat with its fur bristled and ears flattened against its head. The cat's eyes are narrowed and glowing, mouth slightly open in a hiss. It has thick, long fur that is mostly black with some orange stripes. The background shows a stormy sky through a window, enhancing the dramatic mood. The cat is on a dark rug, which contrasts with its bright, intense eyes.

(d) A cartoon-style image of a cat with a slightly annoyed expression. The cat has fluffy white fur, bright green eyes, and its ears are slightly pulled back. It is sitting comfortably on a soft blue cushion, with its tail flicking lightly to the side. The background is a simple, soft yellow to keep the focus on the cat's expression.

(e) A cartoon-style image of a cat with an angry expression. The cat has fluffy grey fur, narrowed yellow eyes, and its ears are flat against its head. It is standing with an arched back on a wooden floor, with its fur bristling and its claws slightly out. The background is a dimly lit room, adding to the menacing atmosphere around the cat.

(f) A cartoon-style image of a cat with a furious expression. The cat has short black fur, glaring red eyes, and its ears are pinned back. It is hissing aggressively, showing sharp teeth, with a puffed tail. The cat stands on a stormy night background, with lightning in the sky, enhancing the dramatic and intense mood.

C Preference Score Calculation

Definition. The Preference Score q evaluates the quality of a sequence of three images with attribute scores a_1, a_2, and $a_3 \in [0, 1]$, and is computed as:

$$q = -\left(a_1 - a_3 + \left|a_2 - \frac{a_1 + a_3}{2}\right|\right)$$

Properties. The Preference Score reflects two aspects:

1. **Order Consistency:** A correctly ordered sequence ($a_1 < a_2 < a_3$) yields a higher q value, while a reversed sequence results in a lower q value.
2. **Distribution Evenness:** A sequence where a_2 is closer to the midpoint between a_1 and a_3 maximizes the score.

Example Calculation. Consider four sequences of attribute scores:

- Sequence $\mathbf{a} = [1, 0.5, 0]$
- Sequence $\mathbf{b} = [0, 1, 0.9]$
- Sequence $\mathbf{c} = [0, 0.5, 1]$

For $\mathbf{a}$:

$$q_a = -\left(1 - 0 + \left|0.5 - \frac{1+0}{2}\right|\right) = -1$$

For $\mathbf{b}$:

$$q_b = -\left(0 - 0.9 + \left|1 - \frac{0+0.9}{2}\right|\right) = 0.35$$

For $\mathbf{c}$:

$$q_c = -\left(0 - 1 + \left|0.5 - \frac{0+1}{2}\right|\right) = 1$$

Since $q_a < q_b$, sequence $\mathbf{b}$ is preferred between sequence $\mathbf{a}$ and $\mathbf{b}$. Sequence $\mathbf{c}$ is preferred between sequence $\mathbf{b}$ and $\mathbf{c}$. This illustrates how the Preference Score penalizes uneven intermediate distributions or incorrect orderings.

D Prompt Styles and Examples

To ensure diversity in user prompts, we utilize four distinct prompt styles inspired by (Qin et al. 2024). Each style varies in how it frames the objective for image generation. For illustration, all the examples below are based on the attribute ”bright,” showcasing how this attribute can be expressed in different styles.

D.1 Prompt Styles

Prompt-Based Style. This style of prompt directly describes the objective to be generated. It provides a clear and concise target for the system. For example:

- ”A series of images showing a garden with increasing brightness, from dawn to midday.”

Instruction-Based Style. This style uses instructional language to explicitly direct the system on what to generate. The phrasing is structured as a command or directive. For example:

- ”Generate a series of images of a morning scene, increasing the brightness and cheerfulness.”

Inspiration-Based Style. This style reflects a user’s desire or inspiration for what they want to see. The prompt is expressed as a personal request or imaginative wish. For example:

- ”I want to see a series of images of a mountain as it gets progressively brighter.”

Hypothesis-Based Style. This style frames the generation task as a hypothetical scenario, often using conditional or reasoning-based language. The prompt includes both the condition and the desired outcome. For example:

- ”If the scene becomes brighter, the series of images will show progressively more illuminated scenes.”

E Examples of Input and Output of Experiments

This appendix presents illustrative examples of the system’s inputs and outputs. Each example consists of a user prompt as input and a sequence of three images as the output. Examples include:

- Success cases before training, highlighting potential alignment issues in untrained systems.
- Failed cases before training, showcasing issues with order consistency or uneven attribute progression.
- Improved cases after training, demonstrating the effectiveness of the proposed SysDPO alignment method.

E.1 Success Case Before Training

Input Prompt: ”I want to see a series of images of a beach, with varying intensity of sunlight from overcast to bright sun.”
Attribute: Sunny
Intermediate Captions:

(a) Generate an image of a serene beach on a cloudy day, with the sun hidden behind thick grey clouds, casting a soft, diffused light on the sand and sea. The waves are calm, and a few seagulls are flying overhead.

(b) Create an image of the same beach, but with the sun breaking through the clouds, casting a warm, golden light on the sand and sea. The waves are slightly choppy, and a few beach umbrellas and sun loungers are scattered along the shore.

(c) Produce an image of the beach on a bright, sunny day, with the sun high in the sky and shining brightly down on the sand and sea. The waves are crashing against the shore, and a group of people are enjoying the day, with some playing beach volleyball and others swimming in the ocean.

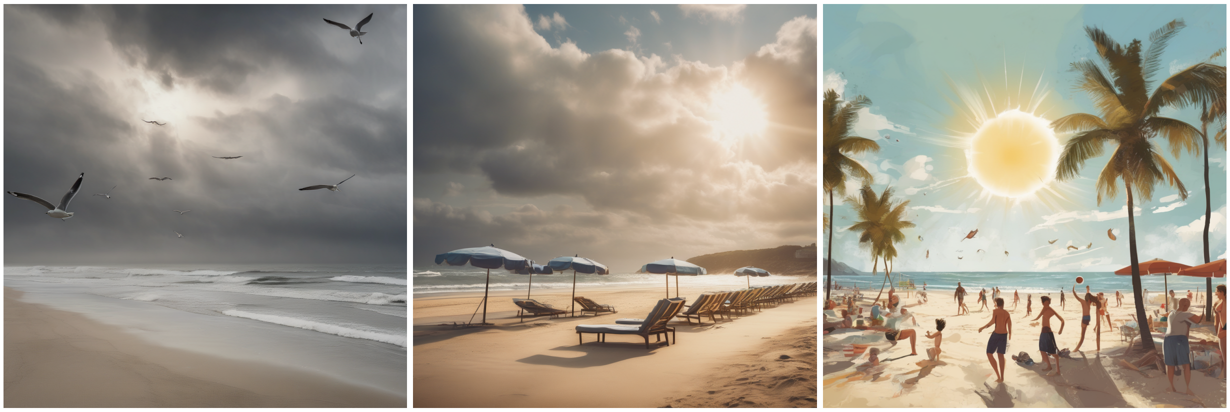


Figure 3: An example of a successful output before training.

E.2 Failed Case Before Training

Input Prompt: ”I want to see a series of images of a lake as the ice increases.”
Attribute: Ice
Intermediate Captions:

(a) Generate an image of a serene lake on a warm summer day, with calm water and lush greenery surrounding it, but with no signs of ice or frost.
(b) Create an image of the same lake but with a thin layer of ice starting to form on the surface, with ripples and imperfections, and the surrounding landscape showing subtle signs of autumnal change, such as fallen leaves and fading greenery.
(c) Produce an image of the lake completely frozen, with a thick layer of ice covering its surface, and the surrounding landscape transformed into a winter wonderland, with snow-covered trees, frozen waterfalls, and a crisp, blue-gray sky.

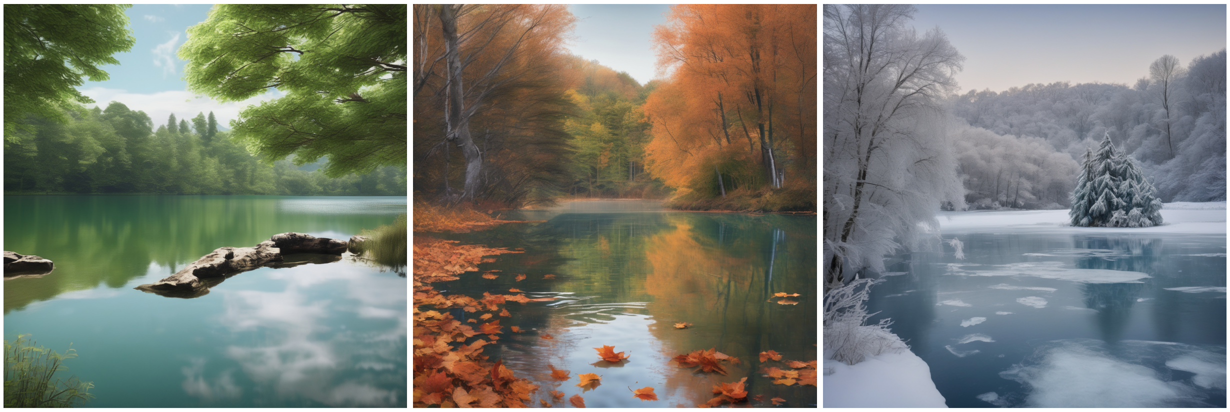


Figure 4: An example of a failure case before training. The first and second images lack clear progression in ice intensity.

E.3 Improved Case After Training

Input Prompt: ”Create a series of images of a lake, increasing the thickness of ice.”
Attribute: Ice
Intermediate Captions:

(a) Generate a serene summer day with a calm and clear lake shoreline, featuring lush green trees and blooming vegetation, with water as the main focus. The water should be smooth and reflective, with a few boats or sailboats in the background. Ignore the water conditions.
(b) Continue with the same shoreline and surroundings, but with the onset of autumn, add gentle winds and decreasing temperatures. The vegetation should be dormant, and while the water is still calm, start to notice a slight decrease in clarity. There should be no ice yet, but hint at the changing conditions with minimal foliage or bare trees along the shore.
(c) In the dead of winter, depict a frozen landscape with the same shoreline, but now with thick, rugged ice covering the entire lake. The trees should be bare and shown as silhouettes against the harsh winter sky. The goal is a highly realistic, textured ice surface with no signs of water, and incorporate any relevant winter elements like snow-covered ground or bare branches.

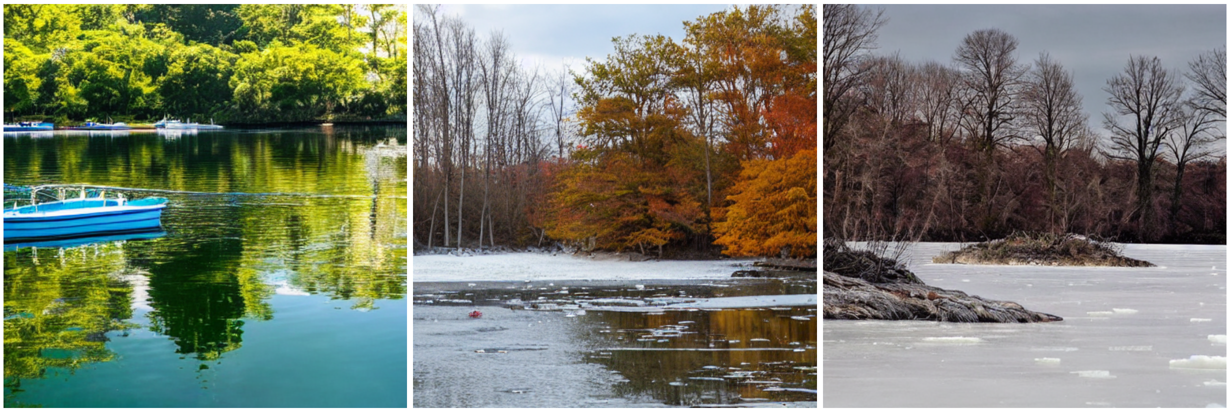


Figure 5: An example of an improved case after training. The sequence shows smooth and consistent progression in the ice intensity.