Dimple: Discrete Diffusion Multimodal Large Language Model with Parallel Decoding

Runpeng Yu Xinyin Ma Xinchao Wang[†]

National University of Singapore {r.yu, maxinyin}@u.nus.edu xinchao@nus.edu.sg

Abstract

In this work, we propose Dimple, the first Discrete Diffusion Multimodal Large Language Model (DMLLM). We observe that training with a purely discrete diffusion approach leads to significant training instability, suboptimal performance, and severe length bias issues. To address these challenges, we design a novel training paradigm that combines an initial autoregressive phase with a subsequent diffusion phase. This approach yields the Dimple-7B model, trained on the same dataset and using a similar training pipeline as LLaVA-NEXT. Dimple-7B ultimately surpasses LLaVA-NEXT in performance by 3.9%, demonstrating that DMLLM can achieve performance comparable to that of autoregressive models. To improve inference efficiency, we propose a decoding strategy termed confident decoding, which dynamically adjusts the number of tokens generated at each step, significantly reducing the number of generation iterations. In autoregressive models, the number of forward iterations during generation equals the response length. With confident decoding, however, the number of iterations needed by Dimple is even only response length. We also re-implement the prefilling technique in autoregressive models and demonstrate that it does not significantly impact performance on most benchmark evaluations, while offering a speedup of 1.5x to 7x. Additionally, we explore Dimple's capability to precisely control its response using structure priors. These priors enable structured responses in a manner distinct from instruction-based or chain-of-thought prompting, and allow fine-grained control over response format and length, which is difficult to achieve in autoregressive models. Overall, this work validates the feasibility and advantages of DMLLM and enhances its inference efficiency and controllability. Code and models are available at https://github.com/yu-rp/Dimple.

Code https://github.com/yu-rp/Dimple

Model https://huggingface.co/rp-yu/Dimple-7B

Playground https://huggingface.co/spaces/rp-yu/Dimple-7B

1 Introduction

Recent months have witnessed a surge of interest in applying diffusion models to natural language processing tasks. While diffusion-based approaches were originally developed for continuous data such as images, recent work has successfully adapted diffusion processes to discrete language modeling. These Discrete Diffusion Language Models (DLMs) reframe generation as a denoising process, allowing parallel decoding and improved control over output structure.

Compared to autoregressive language models, diffusion language models offer several distinct advantages. First, the diffusion process initializes the entire response sequence, providing flexibility and controllability with explicit control over token positions, semantic structures, and output formats.

Preprint. Under review.

Second, the bidirectional attention mechanism enables improved infilling and enhanced planning capabilities. [10, 39] Third, the non-sequential generation mechanism demonstrate potential for faster sampling.

Concurrently, the field of multimodal large language models (MLLMs) has advanced rapidly, with autoregressive architectures such as LLaVA [15], Qwen-VL [4], and InternVL [30] achieving state-of-the-art results on a range of vision-language benchmarks. However, all existing MLLMs rely exclusively on autoregressive generation mechanisms.

To date, no prior work has explored the integration of DLM into the multimodal setting. In this work, we introduce **Dimple**, the first *Discrete Diffusion Multimodal Large Language Model* (DMLLM), which combines a vision encoder with a discrete DLM backbone.

Dimple is trained using a novel two-phase paradigm—**Autoregressive-then-Diffusion**. In the first phase, we adopt standard autoregressive alignment and instruction tuning to align vision and language modalities and tune the instruction following abilities. In the second phase, we switch to diffusion-based masked language modeling to recover the model's parallel decoding capabilities. This hybrid strategy addresses several core challenges of discrete diffusion training, including instability, suboptimal supervision, and length bias.

To further enhance inference efficiency and controllability, we propose **Confident Decoding**, which dynamically adjusts the number of tokens decoded per iteration based on a confidence threshold, and we re-implement the **Prefilling** in autoregressive models. These methods jointly reduce generation complexity. We also explore the **Structure Prior** technique, which enable fine-grained control over response structure.

Experiments over MLLM benchmarks show that Dimple performs comparable to baselines such as LLaVA-NEXT under similar training budget. Moreover, Dimple exhibits unique behaviors during generation, including early answer prediction, structured reasoning control, and precise output formatting, which can be difficult to achieve in autoregressive models.

Contributions. Our key contributions are as follows:

- We introduce **Dimple**, the first discrete DMLLM, and demonstrate that it achieves performance on par with autoregressive MLLMs under equivalent training budgets.
- We design an effective hybrid training paradigm—Autoregressive-then-Diffusion—that mitigates
 the inefficiencies of discrete diffusion training.
- We propose **Confident Decoding**, re-implement **Prefilling**, and explore **Structure Prior** for Dimple inference that improve the efficiency, flexibility, and controllability.

2 Preliminary

2.1 Diffusion Language Models

Diffusion Language Models (DLMs) are a class of generative models that conceptualize text generation as a denoising process over discrete time steps. Let $x_0 \sim p_{\text{data}}(x_0)$ denote the original token sequence, and x_t its noisy version at time $t \in [0,T]$. The forward noising process is defined as a Markov chain $q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$, progressively adding noise to x_0 . For discrete token modeling with an absorbing state, this is formalized as:

$$q(x_t|x_{t-1}) = \mathbf{Cat}(x_t; Q_t^{\top} x_{t-1}), \quad q(x_t|x_0) = \alpha_t x_0 + (1 - \alpha_t) \mathbf{m}, \quad \alpha_t = \prod_{i=1}^t (1 - \beta_i), \quad (1)$$

where $Q_t = (1 - \beta_t)I + \beta_t \mathbf{1m}^{\top}$, with \mathbf{m} being the one-hot representation of a special [MASK] token. The reverse (generative) process is modeled as:

$$p_{\theta}(x_{0:T}) = p_{\theta}(x_T) \prod_{t=1}^{T} p_{\theta}(x_{t-1}|x_t) = \prod_{t=1}^{T} q(x_{t-1}|x_0) p_{\theta}(x_0|x_t),$$
 (2)

where each transition $p_{\theta}(x_{t-1}|x_t)$ is learned to approximate the reverse of the noising process with bidirectional attention. The loss used for training, derived from ELBO, is a reweighted cross-entropy

loss:

$$\mathcal{L}_D = \mathbb{E}_t[\mathcal{L}_t], \quad \mathcal{L}_t = \frac{1}{t} \mathbb{E}_{q(x_t|x_0)} \left[-\sum_{n=1}^N \delta_{x_t^n, \mathbf{m}}(x_0^n)^\top \log f_{\theta}(x_t)^n \right], \tag{3}$$

where $\delta_{x_{+}^{n},\mathbf{m}}$ is an indicator that the token is masked, and f_{θ} denotes the model's prediction.

2.2 Autoregressive Multimodal Models

LLaVA [21] exemplifies the autoregressive approach extended to multimodal settings. It combines a frozen visual encoder (e.g., CLIP ViT-L/14) with a pre-trained language decoder (e.g., Vicuna), connected through a trainable projection layer that maps visual features to the LLM token embedding space. The model is trained to minimize the negative log-likelihood of the target response, following a standard next token prediction loss with caucal attention.

3 Vision Discrete Diffusion Tuning

The task is to combine a pre-trained vision encoder with a discrete diffusion Large Language Model to create a Discrete Diffusion Multimodal Large Language Model, resembling the training paradigm of autoregressive MLLMs [3, 8, 2, 21].

3.1 Inefficiencies of Discrete Diffusion Training

Discrete diffusion training employs bidirectional attention and a timestep-dependent masked language modeling loss, with the timestep randomly sampled during training. However, this training approach introduces two significant issues:

- 1. Compared to next-token prediction loss, masked language modeling has lower utilization of the training corpus. Let x denote an input text sequence used during training, where the total length L is the sum of the prompt length L_{prompt} and the answer length L_{answer} . With the next token prediction loss used in the autoregressive training, each token in the answer part of x receives a supervision signal, enabling the language model to learn from all the knowledge present in the text. In contrast, for each sample in diffusion training, a time step $t \in (0,1]$ is sampled, and the loss is only computed on the masked tokens. This reduces the coverage of the supervision signal, missing part of tokens in corpus.
- 2. In diffusion training, the supervision signal failing to cover the entire generation process. Autoregressive generation produces text of length $L_{\rm answer}$ with $L_{\rm answer}$ steps. In autoregressive training, next-token prediction with causal attention masking ensures that, for each sample, the supervision signal covers every step of the generation process. In contrast, for each sample, the diffusion training provides supervision only for one timestep in the generation process.

These two inefficiencies result in a higher loss, the training instabilities, and, ultimately, poorer performance, if a pure diffusion training strategy is used. Empirical results are discussed in Sec. 4.

3.2 Using Autoregressive Training for Multimodal Ability Learning

To mitigate theses inefficiencies, we break the process of converting a DLM into a DMLLM into two separate stages. In the first stage, we use the existing LLaVA pipeline to train the model to leverage multimodal information in a autoregressive way. In the second stage, we employ the diffusion training to recover the diffusion generation capabilities. The rationale for using autoregressive training on a DLM is twofold:

- 1. Both autoregressive models and absorbing state-based diffusion models can be described as diffusion language models, with the difference lying in the construction of the transition matrix and their respective decoding behaviors. [1] Previous work has demonstrated that an autoregressive language model can be fine-tuned to function as a diffusion language model. [39, 10]
- 2. In our experiments, the model we use, Dream [39], is fine-tuned from Qwen2.5. Thus, it inherently possesses both autoregressive and diffusion generation capabilities. As a result, training with autoregressive loss does not introduce severe inductive biases, which would otherwise impair training or increase the cost of recovering the diffusion generation capabilities.

Furthermore, such autoregressive-then-diffusion training approach also offers performance guarantees and facilitates the seamless transfer of existing MLLM training techniques.

3.3 Training Dimple

The starting point is: Qwen2.5-VL's vision encoder [4], DLM Dream [39], and a randomly initialized two-layer projector, as used in [20].

Phase I: Autoregressive Training In the first phase, we modify the attention mechanism within Dream by replacing the full attention mask with a causal attention mask, enabling the model to operate under an autoregressive decoding paradigm. This phase is designed for vision-language alignment and then visual instruction following learning. We follow the LLaVA-NEXT [22] training recipe and use its datasets.

Phase II: Diffusion Training In the second phase, we transition back to the diffusion generation framework. To this end, we re-enable the full attention mask within Dream. The LLaVA-NEXT instruction-following data is used again. The loss in eq. (3) is used. The mask is only applied to the answer part of each sample.

Generation Length and Token Padding Unlike autoregressive models, DMLLMs generate answers with a given response length and do not rely on a [EOS] token to terminate the generation. To accommodate the diffusion generation process, we replace the [EOS] token in the original training dataset with a random number of padding tokens specific to Dream's tokenizer. These tokens are used by the Dream model to learn to adaptively control the actual length of the generated answers.

3.4 Inference Techniques

Review: Confidence-Based Decoding. In discrete diffusion models, the generation process proceeds iteratively over a fixed number of steps. At each step, a subset of masked positions is selected and updated based on their confidence scores. Let \mathbf{x}_t denote the input sequence at step t. Let $\mathbf{z}_t \in \mathbb{R}^{L \times V}$ be the corresponding logits, where V is the vocabulary size. Taking MaskGIT's decoding algorithm [5] as an example, the decoding process is as follows.

```
Probabilities: \mathbf{p}_t = \operatorname{softmax}(\mathbf{z}_t/\tau); Confidence: c_t^{(i)} = \max(p_t^{(i)}); Select K positions: \mathcal{I}_t = \operatorname{TopK}(c_t, K); Sample tokens x_t^{(i)} \sim \operatorname{Categorical}(p_t^{(i)}) for i \in \mathcal{I}_t.
```

This approach ensures that tokens with higher prediction confidence are updated earlier.

Confident Decoding. In previous confidence-based decoding [5, 39, 31], the number of tokens decoded per step is fixed. However, we argue that decoding should adapt to the semantic structure of the text: some steps may allow many tokens to be confidently predicted, while others may necessitate more caution. We therefore propose **Confident Decoding**, which dynamically adjusts the number of tokens updated per step based on a fixed confidence threshold $\gamma \in (0,1)$. Formally, at each step t, we define $\mathcal{I}_t = \{i \mid c_t^{(i)} \geq \gamma\}$, where $c_t^{(i)}$ is the confidence score for position i. If \mathcal{I}_t is non-empty, all tokens at positions \mathcal{I}_t are decoded. Otherwise, we decode the token at the position with the highest confidence. This method enables:

- decoding multiple tokens simultaneously when model is highly confident, improving efficiency;
- avoiding low-confidence updates, preserving generation quality.

Prefilling. Let L_{prompt} denote the lengths of the prompt, which includes the question, image, and system prompts. Let L_{answer} denote the length of the response. With vision tokens, the prompt can become unignorably long. The use of a full attention mask in diffusion decoding results in a quadratic complexity $\mathcal{O}((L_{\text{prompt}} + L_{\text{answer}})^2)$ per decoding step. To alleviate this cost, we re-implement the **Prefilling** strategy in the autoregressive models, which saves the key-value pairs of the prompt tokens after the first generation step and reuse them in the following steps, reducing the complexity to $\mathcal{O}(L_{\text{answer}}^2)$. But, due to the use of a full attention mask in DMLLM, the prefilling technique is not strictly lossless.

| Model | Dimple-7B (ours) | LLaVA-1.5-7B | LLaVA-NEXT-7B | Eagle-7B | Eagle2-9B | Qwen-VL-7B | Qwen2.5-VL-7B |
|------------------------|------------------|--------------|---------------|----------|-----------|------------|---------------|
| #Training Samples | 1.3M | 1.2M | 1.3M | 2.4M | 27.8M | 1.5B | - |
| #Training Tokens | 0.8B | - | - | - | - | - | 2.6T |
| Base LLM | Dream(Qwen2.5) | Vicuna | Vicuna-1.5 | Vicuna | Qwen2.5 | Qwen | Qwen2.5 |
| GQA [13] | 59.2 | 62.0 | 64.8 | 64.9 | - | 59.3 | - |
| MMBench_en_test[23] | 74.6 | 64.3 | 68.7 | 68.4 | - | - | 83.5 |
| MME_percpetion | 1514 | 1510 | 1519 | 1528 | - | - | - |
| MME_cognition | 432 | - | 332 | - | - | - | - |
| MME_total[9] | 1946 | - | 1851 | - | - | - | 2347 |
| POPE[17] | 86.2 | 85.8 | 86.7 | 88.8 | - | - | - |
| MMMU_val[42] | 45.2 | - | 35.8 | 36.3 | 56.1 | - | 58.6 |
| SQA_img[26] | 77.1 | 66.8 | 72.8 | 70.0 | - | - | - |
| AI2D[14] | 74.4 | - | 65.4 | - | 83.9 | 62.3 | 83.9 |
| ChartQA[29] | 63.4 | - | 54.9 | 67.7 | 86.4 | 65.7 | 87.3 |
| TextQA_eval[35] | 61.6 | - | 64.8 | - | 83.0 | - | - |
| OCRBench[24] | 565 | - | 490 | 529 | - | - | - |
| MathVista_test_mini[27 |] 42.3 | - | 33.0 | - | 63.8 | 37.0 | 68.2 |
| MMVet[40] | 41.2 | 31.1 | 47.3 | - | 62.2 | - | 67.1 |

Table 1: Benchmark Performance Comparison. Dimple outperforms autoregressive models trained on similar data scales but still lags behind SOTA models due to using less than 1/20th of their training data. In the table, the highlighted values in each row indicate the best-performing results among the following five models: Dimple, LLaVA-1.5, LLaVA-NEXT, Eagle, and Qwen-VL, which have a comparable number of training samples or were released around the same time.

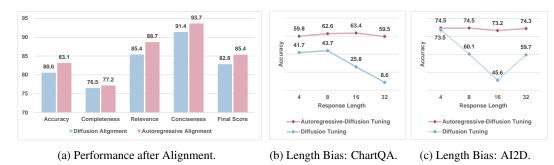


Figure 1: Comparison of Captioning Capabilities After Alignment (Fig. 1a), and the Length Bias Phenomenon on the ChartQA and AI2D Datasets After Instruction Tuning (Figs. 1b and 1c).

4 Experiments

4.1 Benchmark Performance

These experiments aim to demonstrate that the proposed DMLLM Dimple, can match the performance of autoregressive models in multimodal instruction-following tasks, under similar training budgets.

Baselines. We construct baselines comparable to Dimple based on three aspects: model size, language model capability, and the scale and coverage of training data. As Dimple uses the training data of LLaVA-NEXT [22], we include LLaVA-1.5-7B [20] and LLaVA-NEXT-7B [22] as baselines. We also include Eagle-7B [2] and Qwen-VL-7B [3], which is either trained on a similar amount of data or contemporaneous with LLaVA-1.5 and LLaVA-NEXT. These four models share similar training data characteristics with Dimple but use weaker LLMs, forming a theoretical lower-bound reference for Dimple's performance. Since Dimple uses the Dream [39] as LLM, which is fine-tuned from Qwen2.5, we also include Eagle2-9B [19] and Qwen2.5-VL-7B [4]. Both use Qwen2.5 as their LLM. However, these two models are trained on significantly larger datasets—more than 20 times the size of Dimple's training data. Therefore, they serve as a theoretical upper-bound reference for Dimple's performance. Overall, considering the impact of training data on instruction-following and visual understanding capabilities, LLaVA-NEXT can be regarded as the closest baseline to Dimple.

| | | TextVQA | SQA | MMMU | MM | 1E | POPE | MMVet | ChartQA | AI2D |
|---|-------------------------|---------|------|------|------|-----|------|-------|---------|------|
| A | DT (lr=5e-6, 1 Epoch) | 38.4 | 70.2 | 45.1 | 1420 | 360 | 83.9 | 35.2 | 40.8 | 71.3 |
| В | DT (lr=5e-6, 2 Epoch) | 36.6 | 70.6 | 44.0 | 1508 | 397 | 84.7 | 34.9 | 43.7 | 73.5 |
| C | AT (lr=2e-5, 1 Epoch) | 49.2 | 79.5 | 38.1 | 1298 | 403 | 84.3 | 32.9 | 53.7 | 70.5 |
| Е | + DT (lr=5e-6, 1 Epoch) | 49.8 | 81.9 | 42.6 | 1483 | 426 | 85.3 | 34.1 | 53.9 | 74.4 |
| D | AT (lr=5e-6, 1 Epoch) | 49.8 | 83.7 | 43.3 | 1404 | 422 | 82.6 | 36.3 | 55.6 | 71.8 |
| F | + DT (lr=5e-7, 1 Epoch) |) 49.4 | 84.4 | 45.2 | 1514 | 432 | 85.1 | 41.2 | 58.4 | 74.4 |

Table 2: Performance Comparison Between Autoregressive Tuning (AT) and Diffusion Tuning (DT). The results show that the best performance is achieved by first applying autoregressive tuning followed by diffusion tuning.

Numbers in Tab. 1. Dimple's performance is obtained using lmms-eval library, which is also used in LLaVA [16, 44]. For LLaVA-NEXT, results on OCRBench are taken from [2]; results on MathVista test-mini, ChartQA, and AI2D are obtained using lmms-eval. Other results for LLaVA-NEXT and the results for other models are collected from their official reports.

Results and Analysis. Dimple achieves an average performance of 62.4% across all benchmarks, outperforming LLaVA-NEXT, which achieves an average of 58.5%, by 3.9%. Among models trained on comparable datasets, Dimple achieves the best performance on 8 out of 13 benchmarks. However, there remains a gap between Dimple and Qwen2.5-VL or Eagle2. These results align with our intuition: on one hand, Dimple benefits from a stronger language model compared to LLaVA-NEXT and Eagle, leading to better performance with similar training data. On the other hand, due to the limited scale of its training data, Dimple still falls short of models like Qwen2.5-VL and Eagle2, despite leveraging the same base LLM. Nevertheless, the results demonstrate that DMLLM can achieve performance comparable to autoregressive models on general instruction following and visual understanding benchmarks.

4.2 Ablation on Autoregressive Training

Following experiments demonstrates the advantages of the proposed autoregressive-then-diffusion hybrid training paradigm over the pure diffusion training. The pure diffusion pipeline consists of 2 phases: Diffusion Alignment (DA) and Diffusion Tuning (DT). Our hybrid approach incorporates 3 phases: Autoregressive Alignment (AA), Autoregressive Tuning (AT), and Diffusion Tuning (DT).

After Alignment To evaluate the captioning capability of the aligned models, we randomly sampled 128 images from the LLaVA-CC3M dataset [21] (excluded from training) and generated captions using the aligned models. The model trained with DA used the diffusion generation pipeline, while the model trained with AA used the autoregressive generation pipeline. Given an image and its caption, we prompted GPT-4o-2024-11-20 to rate the captions on four dimensions—accuracy, completeness, relevance, and conciseness—on a scale of 1–100, with the final score being the average of the four dimensions. The results, summarized in Fig. 1a, show that the model aligned using AA outperforms the one aligned using DA across all metrics.

After Instruction Tuning We compared six different instruction tuning strategies:

- Strategies A, B: Continue with DT for 1 or 2 epochs after DA.
- Strategies C, D: Perform 1 epoch of AT with two different learning rates after AA.
- Strategies E, F: Continue with 1 epoch of DT using two different learning rates after C and D.

Tab. 2 presents the performance of all six strategies. Since the final goal is to obtain a model capable of diffusion generation, all models were evaluated using the diffusion generation pipeline. First, comparing strategies A and D, which are both trained with the same learning rate for 1 epoch, AT outperforms DT on 7 out of 9 benchmarks. However, models trained with AT alone suffer from a gap between autoregressive training and diffusion-based inference. Then, comparing strategies B and F, the AT-then-DT strategy outperform the pure diffusion tuning strategy across all benchmarks.

¹Metrics on MME and OCRBench are rescaled when calculating the average score.

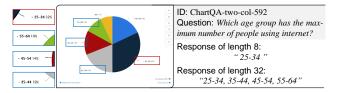


Figure 2: An example of Length Bias Phenomenon Collected from ChartQA. The response is generated by the model after 2-epoch diffusion tuning. The red and blue boxes on the left indicate the source locations of the numbers in the response.

| R.L. | Prefilling | Metric | B.S. | SQA | MMMU | MME | POPE | ChartQA | AI2D |
|------|------------|--------|---------|-------------------------------------|---------------|-----------------------|------------------------------|----------------------|---------------|
| 4 — | × | ACC | 1 or 32 | 84.4 | 43.4 | 1946 | 85.0 | 51.7 | 74.5 |
| | | TPS | 1 | 34.8 | 21.3 | 22.7 | 22.4 | 21.5 | 22.9 |
| | | 113 | 32 | 45.9 | 35.5 | 37.0 | 36.9 | 36.3 | 37.2 |
| | | ACC | 1 or 32 | 84.4 | 43.4 | 1822 | 85.1 | 51.3 | 74.4 |
| | ✓ | TPS | 1 | 49.3 (×1.32) | 19.9 (×1.67) | 21.3 (×1.63) | 20.7 (×1.65) | 20.1 (×1.69) | 21.5 (×1.62) |
| | | | 32 | $168.5 \ \underline{(\times 3.42)}$ | 62.8 (×3.16) | 77.6 (×3.64) | $78.2 \times (3.78)$ | $74.4 \times (3.70)$ | 79.7 (×3.71) |
| | | ACC | 1 or 32 | 84.1 | 45.3 | 1900 | 84.8 | 58.4 | 74.5 |
| | × | TPS | 1 | 29.2 | 19.3 | 19.3 | 19.5 | 19.1 | 19.7 |
| | | 113 | 32 | 44.9 | 39.7 | 40.6 | 40.4 | 40.4 | 40.7 |
| 8 | √ | ACC | 1 or 32 | 84.0 | 45.2 | 1905 | 84.0 | 58.4 | 74.2 |
| | | TPS | 1 | 49.2 (×1.54) | 18.1 (×2.06) | 21.4×2.10 | $20.7 \times (2.07)$ | $20.0 \times (2.12)$ | 21.6 (×2.07) |
| | | 11.5 | 32 | $288.2 \times (\times 5.86)$ | 116.5 (×6.44) | 145.1 $(\times 6.80)$ | $147.4 \times (\times 7.12)$ | 138.9 (×6.95) | 147.9 (×6.85) |

Table 3: Ablation Study on the Prefilling Technique. We compared model performance (ACC) and inference speed (Tokens Per Second, TPS) with and without the use of prefilling. Performance-related rows are highlighted in <code>gray</code>. The underlined values indicate the speedup factor achieved with prefilling. "R.L." and "B.S." denote Response Length and Batch Size, respectively. The results demonstrate that although, in theory, prefilling is not lossless in DMLLMs, it causes minor drops in performance on most benchmarks while delivering substantial improvements in inference speed.

Length Bias Another notable weakness of models trained purely with diffusion is the presence of severe length bias, making them highly sensitive to the response length hyperparameter. This issue is especially pronounced in chart-based QA tasks. Figs. 1b and 1c illustrate the relationship between response length and accuracy on AI2D and ChartQA. The AT+DT model maintains relatively stable performance across varying lengths, whereas the pure diffusion model shows dramatic performance drops as length increases. For example, in ChartQA, accuracy drops from 42.7% to 8.6% as the response length increases from 8 to 32. Fig. 2 shows an example from ChartQA, where the pure diffusion model exhibits a tendency to translate all mask tokens into normal textual tokens. This behavior leads the model to continuing to generate text even after producing a correct answer, as it lacks the ability to "stop" appropriately.

4.3 Ablation on Prefilling

The following experiments investigate the impact of the prefilling technique on both performance and generation speed in DMLLMs. We conducted experiments with response lengths of 4 and 8, evaluating benchmark performance with and without prefilling. As shown in Tab. 3, prefilling does not lead to significant performance degradation on most datasets, with the average performance drop across all benchmarks being only 0.8%. This indicates that, in the current model, visual perception and the utilization of image tokens remain largely unchanged during text generation. Next, we examined the effect of prefilling on generation speed. By adjusting the batch size to 1 and 32, we simulated scenarios of low and high GPU utilization, respectively. The results show that:

- Under low GPU utilization (batch size = 1), prefilling yields an average speedup of 1.79×.
- Under high GPU utilization (batch size = 32), the acceleration effect is more pronounced, reaching up to 7×.

Question: What is the common item in the two images? **Response**: In the first image, there is a pair of black scissors. In the second image, there is a pair of black scissors and a pink plate with a pig face on it. The common item in the two images is the scissors. Response Length / #Remaining Tokens / Actual Iterations: **Generation History:** In there is black the the first image a pair of scissors In 2 1 4 5 16 5 4 second of black image there is pair scissors and pink plate 2 2 13 12 15 14 17 18 19 23 with face it The item in the images is on common two pig 25 27 4 5 22 24 26 20 21 the scissors 11 10 9

Table 4: Example: Structured Reasoning and Early Answering.

 Question: Extract the information in the image.

 Response: {"Date: "Apr 18, 2018", "Time: "12:00 AM",}

 Response Length/#Remaining Tokens/Actual Iterations: 32/22/7

 Generation History:

 { "Date: "Apr 18, 2018", "Time: "12:00 AM",}

 - - - - - 5 1 1 1 1 1 1 1 1 1 1 1 2 1 - - - 3 3 4 3 4 6 7 -

Table 5: Example: Structured Output.

5 Generation Behaviors

In this section, we provide a showcase to demonstrate that DMLLMs exhibit fundamentally different decoding behaviors compared to autoregressive models. These behavioral differences not only yield improvements in decoding efficiency but also enhance controllability and interpretability during generation.

Unlike autoregressive models, discrete diffusion models offer the unique capability of **explicitly controlling the content** generated at arbitrary positions. This allows us to specify certain tokens as fixed in the final answer prior to generation. We refer to such tokens as **structure priors**.

We present three illustrative examples. Each includes the input image, the question, and the generated response from Dimple. For each case, we annotate: the **response length** (i.e., the total number of tokens in the final answer), the **number of remaining tokens** that need to be generated after applying the structure prior, and the **actual number of decoding steps** (actual iteration) used in generation. Furthermore, we visualize the **generation history**, which records at which iteration each token was generated. Tokens provided by structure priors are marked with "-", and special tokens (e.g., padding) are excluded from the generation history but counted in the actual number of decoding steps. To enhance interpretability, we colored the response: tokens decoded earlier are shown in blue, while those generated later appear in red. Tokens with similar decoding steps share similar colors. All examples leverage the Prefilling technique and proposed Confident Decoding strategy.

5.1 Example 1 (Tab. 4): A New Paradigm for Structured Reasoning and Early Answering

In this example, the input consists of two images, and the model is asked to identify the common object present in both. We inject three structure priors into the expected answer: "In the first image, there", "In the second image, there" and "The common item in the two images is".

These priors guide the model to follow a reasoning trajectory resembling structured thought, going beyond conventional instruction prompts or chain-of-thought (CoT) strategies used in autoregressive

Most preferred objects of different **Ouestion**: How many objects are preferred by more than 90 percent of 100 people in at least one category? Please think step by step. Response: Nothing is preferred by more than 90 percent of people in any category. Thus, the answer is \box{Zero}. Response Length/#Remaining Tokens/Actual Iterations: 32/23/16 **Generation History:** Nothing 0 preferred more than percent people category 10 10 12 11 3 17 16 15 14 13 10 8 6 5 is box Zero the answer 2 **Response**: All of the objects in the chart have a maximum of 90 percent, but no object actually reaches 90. Therefore, none of the objects are preferred by more than 90 percent of people in at least one category. Thus, the answer is \box{Zero}. Response Length/#Remaining Tokens/Actual Iterations: 64/55/37 **Generation History:** objects in the chart have maximum of percent but 22 23 25 29 20 17 15 19 28 24 27 33 16 18 38 21 26 object actually reaches Therefore the objects preferred none 34 35 36 37 31 32 30 13 14 12 11 11 10 8 7 6 by more 9 0 percent people in at least one category Thus 6 6 5 5 5 5 4 5 6 6 6 6 6 6 the answer is box Zero 3

Table 6: Example: Length Control.

models. Unlike those paradigms, which rely on indirect guidance, structure prior enables direct and precise control over intermediate reasoning steps.

Remarkably, the correct answer "scissors" is decoded at the **10**th iteration, prior to the completion of the full response. The subsequent steps focus on completing the thinking trajectory. This illustrates a key strength of diffusion models: they can arrive at the final answer earlier in the generation process. In contrast, autoregressive models must complete the full sequence before producing the answer.

5.2 Example 2 (Tab. 5): Structured Output

In this example, the task involves extracting visual information from an image. The question does not specify the desired output format and the attributes required. All of these are indicated by the structure prior. In this example, the structure prior is "{date:", "time:" and "}". These priors define a JSON-like response layout, and the model successfully generates structured content accordingly.

This example also provides direct evidence of the effectiveness of **Confident Decoding**. The number of tokens decoded per iteration varies dynamically. For instance, 9 tokens corresponding to the date and year are decoded simultaneously in a single step. Despite a total of 22 tokens to be generated, the model completes generation in only 7 iterations, one-third of the total number of tokens. This parallelism significantly accelerates inference.

5.3 Example 3 (Tab. 6): Length Control

Controlling output length is difficult in autoregressive generation. However, discrete diffusion models inherently support this capability. In this case, we specify structure priors at the end of the sequence. Specifically, we force tokens at positions [-12:-4] to be: "Thus, the answer is $box{}$ ". This allows us to strictly control where the model concludes its generation and places the final answer. We demonstrate two configurations with response lengths of 16 and 32. In each case, the model adjusts its reasoning span to fill the available token budget appropriately. This example confirms the **fine-grained length controllability** afforded by diffusion-based decoding—a powerful advantage for applications requiring concise or bounded outputs.

6 Conclusion

In this work, we introduce **Dimple**, the first Discrete Diffusion Multimodal Large Language Model (DMLLM). Through a hybrid training strategy and specialized inference methods, we demonstrate that DMLLM can match the performance of autoregressive models while offering unique advantages such as parallel decoding and structure-aware generation. Our experiments validate the feasibility and strengths of the DMLLM paradigm, opening up new directions for efficient and controllable multimodal generation.

Supplementary Material

S1 Training and Evaluation

S1.1 Training Configuration and Hyperparameters

| Training Phase | 1st: Autoregressive Alignment | 2nd: Autoregressive Instruction Tuning | 3rd: Diffusion Instruction Tuning | | | |
|-----------------|-------------------------------|--|-----------------------------------|--|--|--|
| Training Method | Autoregressive | Autoregressive | Diffusion | | | |
| Training Data | LLaVA-CC3M-Pretrain | LLaVA-NEXT Instruction Tuning | | | | |
| #Training Data | 559k | 739k | | | | |
| learning rate | 0.001 | 5e-6 | 5e-7 | | | |
| batch size | 256 | 128 | 128 | | | |
| weight decay | 0 | 0 | 0 | | | |
| warmup ratio | 0.03 | 0.03 | 0.03 | | | |
| lr schedule | linear | linear | linear | | | |
| max grad norm | 1 | 1 | 0.1 | | | |
| epoch | 1 | 1 | 1 | | | |
| optimizer | AdamW | AdamW | AdamW | | | |

Table S1: Training Configuration and Hyper-Parameters.

The training of Dimple can be divided into three stages. The first two stages involve autoregressive alignment and instruction-following tuning, while the final stage utilizes diffusion-based instruction-following tuning. The specific training configurations and hyperparameters for each stage are detailed in Tab. S1. The training was completed on H100 clusters with a total of approximately 100 GPU hours. Experiments in Tab. 3 is conducted on one single H100 GPU.

S1.2 Training Sample Padding

| Text Input | for Autoregressive Training | | | | |
|-----------------------------------|--|--|--|--|--|
| User: | [BOS] <image/> \nWhat type of item is holding the napkin?\nAnswer the question with GPT-T-COCO format. [EOS] | | | | |
| Assistant: | [BOS] A clear napkin holder is holding the napkin. [EOS] | | | | |
| User: | [BOS] What color is the napkin in the napkin holder? [EOS] | | | | |
| Assistant: | [BOS] The napkin inside the napkin holder is blue-green. [EOS] | | | | |
| Text Input for Diffusion Training | | | | | |
| User: | [BOS] <image/> \nWhat type of item is holding the napkin?\nAnswer the question with GPT-T-COCO format. [EOS] | | | | |
| Assistant: | [BOS] A clear napkin holder is holding the napkin. [Padding] [Padding] | | | | |
| | n_1 [Padding] Tokens | | | | |
| User: | [BOS] What color is the napkin in the napkin holder? [EOS] | | | | |
| Assistant: | [BOS] The napkin inside the napkin holder is blue-green. [Padding] [Padding] | | | | |
| | n_2 [Padding] Tokens | | | | |

Table S2: Training Examples Used for Autoregressive Training and Diffusion Training. n_1 and n_2 n are random integers that enhance the model's robustness to varying response lengths.

The Dream model relies on a predefined response length to determine the length of generated outputs, ensuring that the number of textual tokens does not exceed this length. When the number of textual tokens is less than the response length, Dream generates additional [Padding] tokens so that the total number of tokens equals the predefined response length. Therefore, the model does not rely on a [EOS] token to terminate generation. Instead, [EOS] tokens in the original training data should be replaced with [Padding] tokens.

To enhance the Dimple model's robustness to response length, we randomly replace each single <code>[EOS]</code> token with n <code>[Padding]</code> tokens. The max and min value of n is selected based on the length of the ground truth response. For a ground truth answer with the length of l, the derivation of n can

Caption Evaluation Instructions

You will be given one image and one generated visual caption describing the image. Your task is to carefully evaluate how well the caption reflects the content of the image based on four dimensions:

- 1. Accuracy: Does the caption correctly describe the objects, actions, or scene in the image?
- 2. Completeness: Does the caption capture the essential and important aspects of the image, without omitting critical content?
- 3. Relevance: Is all the information in the caption relevant to the image? Are there any hallucinated, fabricated, or irrelevant elements?
- 4. Conciseness: Is the caption free from redundancy or unnecessary elaboration?

For each dimension, assign a score from 0 to 100 and present it in the following format:

```
**Accuracy**
Justification: {{your explanation here}}
Accuracy score: {{score}}
**Completeness**
Justification: {{your explanation here}}
Completeness score: {{score}}
**Relevance**
Justification: {{your explanation here}}
Relevance score: {{score}}
**Conciseness**
Justification: {{your explanation here}}
Conciseness score: {{score}}
Then, provide the average score over the four dimensions. Justify your overall score in a
paragraph and finish with the sentence:
"The final score is {{score}}."
Image: [image]
Generated caption: [caption]
```

Table S3: Instructions used when Evaluating Captions for the Models after Alignment.

be written as follows.

```
\begin{split} n &\sim \text{RandomInteger}(n_{min}, n_{max}), \\ (n_{min}, n_{max}) &= \begin{cases} (l+1, \ 2^{\lceil \log_2(l+1) \rceil}), & \text{if } n < 16, \\ (l+1, \ 16 \cdot \left\lceil \frac{l+1}{16} \right\rceil), & \text{otherwise} \end{cases} \end{split}
```

Tab. S2 show an example of input sample for diffusion training.

S1.3 Evaluation Configuration

We conducted benchmark evaluations using the lmms-eval library. Both the evaluation metrics and prompts were based on the default configuration provided by lmms-eval. Throughout all evaluations, we employed the MaskGIT decoding algorithm from Dream, with the algorithm temperature set to 0 and confident decoding disabled. The overall decoding temperature was set to 0, and the top-p value was fixed at 0.01, resulting in a largely deterministic generation process.

The most influential parameter affecting evaluation performance is the response length. For all datasets except MMVet, we experimented with response lengths of 4, 8, and 16. Table 1 reports the

Algorithm 1: Confident Decoding

```
// Take temperature as an example of probability adjustment.
// Confidence is calculated in a MaskGIT way.
// Use random selection callback.
Input: Input token sequence \mathbf{x}_t, logits \mathbf{z}_t, temperature \tau, threshold \gamma, number of positions K to
         select if the fallback is triggered, number of masked tokens N
Output: Updated token sequence x_{t+1}
// Step 1: Sample token predictions and compute confidence
// Pre-revision probabilities
\mathbf{p}_t \leftarrow \operatorname{softmax}(\mathbf{z}_t);
// Post-revision probabilities
\tilde{\mathbf{p}}_t \leftarrow \operatorname{softmax}(\mathbf{z}_t/\tau);
foreach i \in \{1, \dots, N\} do
    // MaskGIT confidence
    c^{(i)} = \max(\mathbf{p}_t^{(i)}) ;
    // sampling
  \tilde{x}_{t}^{(i)} \sim \text{Categorical}(\tilde{\mathbf{p}}_{t}^{(i)});
// Step 2: Position selection
x_{t+1} \leftarrow x_t;
if \exists i, c^{(i)} \geq \gamma then
    // Step 2.1: Confident decoding
    \mathcal{I} \leftarrow \{i \mid c^{(i)} \geq \gamma, i \in \{1, \dots, N\}\}\;;
    foreach i \in \mathcal{I} do
      | x_{t+1}^{(i)} \leftarrow \tilde{x}_t^{(i)} ;
else
     // Step 2.2: Fallback: decoding via random selection
    Randomly select K positions: \mathcal{I} \leftarrow \text{RandomSample}(\{1, \dots, N\}, K);
    foreach i \in \mathcal{I} do
       x_{t\perp 1}^{(i)} \leftarrow \tilde{x}_t^{(i)};
return x_{t+1}
```

best results among these three settings. The MMVet dataset is an exception due to the high variability in the ground truth response lengths—ranging from a single word to over 500 tokens. For MMVet, we fixed the token length for all responses to 64. While this is not an optimal choice, as it limits the model's capacity to fully address long-response questions and wastes tokens on short responses, it provides a uniform evaluation setting. We consider adaptive response length selection as a direction for future research. During evaluation, the number of generation steps was set equal to the response length, enforcing the model to decode exactly one token per step.

S1.4 Evaluation of Model after Alignment

To evaluate the captioning capabilities of the model after alignment, we employed GPT-40 for assessment. The prompt used for evaluation is shown in Tab. S3.

S1.5 Confident Decoding

We propose **Confident Decoding**, a token selection strategy designed for diffusion-based generation. Unlike traditional decoding methods that fix the number of tokens to be updated per iteration, Confident Decoding dynamically selects positions to decode based on the model's confidence in its predictions.

Overview. At each decoding step, the model evaluates the confidence of its predictions across all masked positions. If any of the confidence scores exceed a predefined threshold γ , the corresponding

tokens are decoded all together. Otherwise, a fallback strategy is used to ensure generation continues in accordance with the total decoding budget. That is, if no confidence score exceeds the threshold, the second stage of selection is triggered, and another decoding algorithm can be invoked. Thus, the complete decoding pipeline can combine Confident Decoding with MaskGIT, Random selection, or other strategies. Alg. 1 presents an example of the pseudocode for Confident Decoding. This mechanism enables adaptive decoding, leading to both improved generation quality and computational efficiency.

Confidence Estimation. Confident Decoding does not impose any constraints on the method for computing confidence. It can use the maximum prediction probability as the confidence value, as used in MaskGIT, the entropy of the prediction distribution, or the margin between the highest and second-highest prediction probabilities. In general, the confidence can be expressed as a function $c = \phi(p)$, where p is a prediction probability vector of length V, and the output is a scalar confidence score.

Since the prediction probabilities in LLMs can be revised by decoding parameters such as temperature, top-p and top-k, prior work has considered two options for choosing the input probability p: either before [31] or after such revisions [39]. In our experiments, we use the prediction probabilities before these revisions to compute confidence. This choice is motivated by the fact that adjustments, like temperature and top-p, are applied in a position-wise manner, without accounting for interpositional relationships or preserving the relative confidence across different positions. For example, in high-temperature or low top-p settings, the revised probability may become almost deterministic, where one position has a probability of approximately 1 and all others are 0. Such a probability is uninformative for computing confidence, as all positions would yield identical confidence scores.

S2 Related Work

S2.1 Diffusion Language Models

The first problem when introducing diffusion process into language decoding, is the choice of a continuous diffusion space or discrete one. There are explorations in the continuous spaces. DiffuSeq [41] use a partial noising and conditional denoising process to model sequence-to-sequence diffusion in continuous space. Argmax Flows and Multinomial Diffusion propose continuous relaxations for categorical distributions for better generation stability [12]. SED [36] directly use the embedding space for the diffusion of natural language tokens. For the exploration in the discrete space, Structured Denoising Diffusion Models (SDDM) introduce discrete diffusion tailored to linguistic structures [1]; RDM [45] reparameterizes the backward discrete diffusion into a step sampling process to provide more flexible and unified framework. For training diffusion language model, loss function is the key. MDLM [33], MD4 [34] and SEDD [25] show that both a simple weighted mask language modeling loss and a score entropy loss, which is adopted from the score matching loss, are suitable for training. In this work, we follow the thread of masked language modeling scheme for the language diffusion process. In this direction, Diffusion-LM [37] applies diffusion to masked language modeling to enhance controllable text generation; DiffusionBERT [32] uses pretrained BERT as an initialization of the diffusion training to accelerate convergence and improve performance. Besides using a pure diffusion language model, auto-regressive hybrids like AR-Diffusion [18] and Semi-autoregressive methods like SSD-LM [11] aim to combine the benefits of diffusion and autoregressive modeling. SSD-LM also shows that with a simplex projection the diffusion process can also be conducted in the natural vocabulary space. Recently, diffusion language models have been scaled to show comparable performance with the SOTA auto-regressive model, to learn instruction following ability and to have even better planning ability [31, 10, 38, 39].

S2.2 Multimodal Large Language Models

Multimodal large language models (MLLMs) have rapidly progressed with several open-source series such as LLaVA, Eagle, InternVL, and Qwen-VL. LLaVA aligns vision encoders with LLMs via visual instruction tuning and demonstrates competitive performance using efficient training and external tool use [21, 20, 15, 22]. Eagle models explore vision-language integration through multiple encoders and long-context strategies, notably Automatic Degrade Sampling and Image Area Preservation [2, 19, 6]. InternVL focuses on scaling vision-language models, evolving from large

vision encoders to joint multimodal pretraining, and achieves state-of-the-art results with models like InternVL2.5 and InternVL3 [8, 7, 30, 28]. Qwen-VL introduces multilingual vision-language models with dynamic resolution processing and bounding-box generation, achieving competitive performance across diverse tasks [3, 43, 4].

All of the above MLLMs operate under the auto-regressive paradigm. Our work demonstrates that discrete diffusion-based MLLMs not only achieve comparable performance to auto-regressive models but can also operate more efficiently.

S3 Discussions

Limitations. Dimple-7B is trained on less than 1/20th of the data used by state-of-the-art models such as Qwen2.5-VL and Eagle2. Consequently, while Dimple demonstrates competitive performance among models of similar scale, it still falls short compared to the most powerful models. Therefore, scaling up training data and model parameters is a key future direction for DMLLM. Although confident decoding improves efficiency, the full attention mechanism in discrete diffusion still incurs quadratic memory and computational complexity, limiting scalability to longer sequences and low-resource environments. Thus, further investigation into optimizing DMLLM inference is essential.

Broader Impact Dimple's safety concerns arise from both the underlying DLM model and Dimple's vision encoding process. Like other MLLMs, it is also susceptible to issues such as hallucination and generation bias.

Safeguards. The following Usage Disclosure is included in the Dimple document. "We release the Dimple model for academic and non-commercial research purposes only. Users are strongly discouraged from using the model to generate misleading or harmful content, including but not limited to disinformation, impersonation, or harassment. Dimple is not designed or validated for use in safety-critical domains such as medical decision-making, autonomous vehicles, or legal advice systems."

S4 Examples

In this section, we provide additional examples of Dimple's generations to qualitatively showcase its capabilities. A typical phenomenon is that the existence of randomness in the decoding algorithm affects the length of the meaningful text generated. Here, meaningful text refers to tokens excluding special tokens such as padding token and mask token. Given a response length, shorter outputs indicate a higher proportion of meaningless padding tokens, whereas longer outputs imply fewer such tokens. Here, the decoding algorithm specifically refers to the method used to select which tokens are decoded, such as MaskGIT. The randomness in the decoding algorithm is affected by two factors: first, whether token selection is performed via sampling or by greedily choosing the position with the highest confidence; second, the value of the temperature parameter. When there is no randomness, Dimple tends to generate shorter responses. In contrast, with randomness, Dimple is more likely to produce longer responses.

Not all questions exhibited the aforementioned phenomenon. Some questions are inherently unaffected. For instance, as shown in Tab. S4, mathematical reasoning tasks naturally require multi-step reasoning, resulting in relatively long answers regardless of the presence of randomness. In contrast, in Tabs. S5 and S6, tasks such as pure OCR recognition or counting—which involve only perception without the need for deeper understanding or analysis—tend to yield short answers, regardless of the decoding strategy.

However, for tasks where the response length is not intrinsically fixed, the presence or absence of randomness in decoding has a significant impact. For example, in Tabs. S7 to S9, tasks that require both perception and visual understanding, or involve describing images, clearly show that decoding with randomness tends to generate much longer responses.

In Tabs. S7 to S9, the response length is set to be 128 and the decoding algorithm is MaskGIT. For the "without randomness" setup, the temperature of the MaskGIT is 0. For the "with randomness" setup, the temperature is 0.4.

Mathematical Reasoning

$$(x+3)^2 = 4$$

User: First please perform reasoning, and think step by step to provide best answer to the

following question: What is x in the equation?

Dimple: The equation in the image is $(X+3)^2 = 4$. To find the value of x, we need to solve the

equation. First, we take the square root of both sides to get rid of the square. This gives us $x + 3 = \pm 2$. Then, we subtract 3 from both sides to isolate x. This gives us $x = -3 \pm 2$. This results in two possible values for x: x = -3 + 2 = -1 and x = -3 - 2 = -5. So, the

value of x is either -1 or -5.

Table S4: Example of Dimple Generation. The image and question are collected from the MMVet dataset [40]. This example demonstrates Dimple's capabilities in OCR and mathematical reasoning.

OCR



User: What are all the scene text in the image?

Dimple: Connecticut law of 1642, If any man or woman be a witch- that is, hath or consulteth

with a familiar spirit-they shall be put to death

Table S5: Example of Dimple Generation. The image and question are collected from the MMVet dataset [40]. This example demonstrates Dimple's capabilities in pure OCR.

Counting

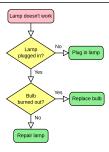


User: How many tomatoes are there?

Dimple: Five

Table S6: Example of Dimple Generation. The image and question are collected from the MMVet dataset [40]. This example demonstrates Dimple's capabilities in counting.

Chart Understanding



User: What is the solution if the lamp is plugged in and the bulb has

burned out?

Dimple (w/o Randomness): The solution is to replace the bulb.

solution would be to replace the bulb. This is because if the bulb is dead and it's not plugged into the power source, then replacing the bulb is necessary to restore the functionality of the lamp.

Table S7: Example of Dimple Generation. The image and question are collected from the MMVet dataset [40]. This example demonstrates Dimple's capabilities in OCR and chart understanding.

Recognition



User: First please perform reasoning, and think step by step to provide best

answer to the following question: Is this airplane taking off or landing?

Dimple (w/o Randomness): The airplane is taking off.

Dimple (w Randomness): The airplane is taking off because, as can be seen in the image, the

landing gear is off the ground, and the lower part of the wing is visible. This indicates the airplane is not yet on the runway, and it is in the air.

Table S8: Example of Dimple Generation. The image and question are collected from the MMVet dataset [40]. This example demonstrates Dimple's capabilities in recognition.

Chicken Nugget Map Meme

Sometimes I just look at pictures of the earth from space and I marvel at how beautiful it all is.



User: Can you explain this meme in detail?

Dimple (w/o Randomness): The meme features a tray of golden chicken nuggets, arranged to resem-

ble a world map.

Dimple (w Randomness): The meme is a humorous image featuring a tray of golden breaded or

fried food on a baking sheet, along with the text "Sometimes I just look at the pictures of the Earth from space and I marvel at how beautiful it is." The meme appears to be comparing the appearance of the food on the tray to a stunning view of a planet from space, emphasizing the beauty of the Earth in relation to the food. The contrast between the food on the tray and the celestial image adds to the comparison in a playful and entertaining way. The overall message of the meme is to highlight

the wonder and beauty of the natural world.

Table S9: Example of Dimple Generation. The image and question are collected from the LLaVA Paper [21]. This is a typical example demonstrates the model's ability to understand memes and the abstract humor embedded within them.

References

- [1] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023.
- [2] Mozhgan Nasr Azadani, James Riddell, Sean Sedwards, and Krzysztof Czarnecki. Leo: Boosting mixture of vision encoders for multimodal large language models, 2025.
- [3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.
- [4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025.
- [5] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer, 2022.
- [6] Guo Chen, Zhiqi Li, Shihao Wang, Jindong Jiang, Yicheng Liu, Lidong Lu, De-An Huang, Wonmin Byeon, Matthieu Le, Tuomas Rintamaki, Tyler Poon, Max Ehrlich, Tuomas Rintamaki, Tyler Poon, Tong Lu, Limin Wang, Bryan Catanzaro, Jan Kautz, Andrew Tao, Zhiding Yu, and Guilin Liu. Eagle 2.5: Boosting long-context post-training for frontier vision-language models, 2025.
- [7] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, Ji Ma, Jiaqi Wang, Xiaoyi Dong, Hang Yan, Hewei Guo, Conghui He, Botian Shi, Zhenjiang Jin, Chao Xu, Bin Wang, Xingjian Wei, Wei Li, Wenjian Zhang, Bo Zhang, Pinlong Cai, Licheng Wen, Xiangchao Yan, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites, 2024.
- [8] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks, 2024.
- [9] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. MME: A comprehensive evaluation benchmark for multimodal large language models. *CoRR*, abs/2306.13394, 2023.
- [10] Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models, 2025.
- [11] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control, 2023.
- [12] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions, 2021.
- [13] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.
- [14] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images, 2016.
- [15] Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. Llava-next: Stronger llms supercharge multimodal capabilities in the wild, 2024.
- [16] Bo Li, Peiyuan Zhang, Kaichen Zhang, Fanyi Pu, Xinrun Du, Yuhao Dong, Haotian Liu, Yuanhan Zhang, Ge Zhang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Accelerating the development of large multimoal models, 2024.
- [17] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

- [18] Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. Diffusion models for non-autoregressive text generation: A survey, 2023.
- [19] Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, Nadine Chang, Karan Sapra, Amala Sanjay Deshmukh, Tuomas Rintamaki, Matthieu Le, Ilia Karmanov, Lukas Voegtle, Philipp Fischer, De-An Huang, Timo Roman, Tong Lu, Jose M. Alvarez, Bryan Catanzaro, Jan Kautz, Andrew Tao, Guilin Liu, and Zhiding Yu. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models, 2025.
- [20] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023.
- [21] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- [22] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024.
- [23] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player? CoRR, abs/2307.06281, 2023.
- [24] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12), 2024.
- [25] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution, 2024.
- [26] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
- [27] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.
- [28] Gen Luo, Xue Yang, Wenhan Dou, Zhaokai Wang, Jiawen Liu, Jifeng Dai, Yu Qiao, and Xizhou Zhu. Mono-internyl: Pushing the boundaries of monolithic multimodal large language models with endogenous visual pre-training, 2025.
- [29] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022.
- [30] Fnu Mohbat and Mohammed J. Zaki. Llava-chef: A multi-modal generative model for food recipes, 2024.
- [31] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025.
- [32] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024.
- [33] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024.
- [34] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025.
- [35] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 8317–8326, 2019.
- [36] Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation, 2022.
- [37] Shitong Xu. Clip-diffusion-lm: Apply diffusion model on image captioning, 2022.
- [38] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning, 2025.

- [39] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025.
- [40] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities, 2023.
- [41] Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. Seqdiffuseq: Text diffusion with encoder-decoder transformers, 2023.
- [42] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv preprint arXiv:2311.16502*, 2023.
- [43] Haoyu Zhang, Yangyang Guo, and Mohan Kankanhalli. Joint vision-language social bias removal for clip, 2024
- [44] Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Reality check on the evaluation of large multimodal models, 2024.
- [45] Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation, 2024.