End-to-End Multi-Modal Diffusion Mamba

Supplementary Material

7. Appendix A

7.1. Theorem 1

Theorem 1. *According to Bayes' theorem and the Gaussian distribution density formula, the following calculation relationship of $\frac{p_{data}(y)}{p_{data}(z^g_{n,t})}$ is obtained:*

$$\frac{p_{data}(y)}{p_{data}(z^g_{n,t})} = \exp\left(\frac{\|z^g_{n,t}\|^2}{2} - \frac{\|z^g_{n,t} - \sqrt{\bar{\alpha}^g_t} z^g_{n,0}\|^2}{2(1-\bar{\alpha}^g_t)}\right) \tag{1}$$

Proof. According to [50], from Bayes' theorem, we express the posterior probability as:

$$p_{data}(z^g_{n,0}|z^g_{n,t}) = \frac{p(z^g_{n,t}|z^g_{n,0}) p_{data}(z^g_{n,0})}{p(z^g_{n,t})}. \tag{2}$$

Rearranging, we obtain:

$$\frac{p_{data}(z^g_{n,0})}{p_{data}(z^g_{n,t})} = \frac{p(z^g_{n,t}|z^g_{n,0})}{p(z^g_{n,t})}. \tag{3}$$

Given the real data $z^g_{n,0}$, the probability of the diffused noise state is $p(z^g_{n,t}|z^g_{n,0})$. $p(z^g_{n,t})$ is the marginal distribution of all possible $z^g_{n,0}$ after diffusion.

The forward noise addition process in the diffusion model is defined as follows:

$$z^g_{n,t} = \sqrt{\bar{\alpha}^g_t} z^g_{n,0} + \sqrt{1-\bar{\alpha}^g_t}\epsilon^g_{n,t}, \epsilon^g_{n,t} \sim \mathcal{N}(0, I), \tag{4}$$

and it can be seen that given $z^g_{n,0}$, $z^g_{n,t}$ obeys the Gaussian distribution:

$$p(z^g_{n,t}|z^g_{n,0}) = \mathcal{N}(z^g_{n,t}; \sqrt{\bar{\alpha}^g_t} z^g_{n,0}, (1-\bar{\alpha}^g_t) I), \tag{5}$$

where this conditional probability indicates that $z^g_{n,t}$ is a Gaussian distribution with $\sqrt{\bar{\alpha}^g_t} z^g_{n,0}$ as mean and $(1-\bar{\alpha}^g_t)I$ as variance.

Then, for the marginal distribution $p(z^g_{n,t})$ can be calculated by integration:

$$p(z^g_{n,t}) = \int p(z^g_{n,t}|z^g_{n,0}) p_{data}(z^g_{n,0}) dz^g_{n,0}. \tag{6}$$

Typically, we assume that the underlying distribution of the data follows a standard Gaussian:

$$p_{data}(z^g_{n,0}) = \mathcal{N}(z^g_{n,0}; 0, I), \tag{7}$$

since the convolution of two Gaussian distributions is a Gaussian distribution, $p(z^g_{n,t})$ is still a Gaussian distribution:

$$p(z^g_{n,t}) = \mathcal{N}(z^g_{n,t}; 0, I). \tag{8}$$

Combining the above derivation, we get:

$$\frac{p_{data}(z^g_{n,0})}{p_{data}(z^g_{n,t})} = \frac{p(z^g_{n,t}|z^g_{n,0})}{p(z^g_{n,t})}. \tag{9}$$

Then, substitute into the Gaussian distribution density formula:

$$\frac{p(z^g_{n,t}|z^g_{n,0})}{p(z^g_{n,t})} = \frac{\exp\left(-\frac{\|z^g_{n,t} - \sqrt{\bar{\alpha}^g_t} z^g_{n,0}\|^2}{2(1-\bar{\alpha}^g_t)}\right)}{\exp\left(-\frac{\|z^g_{n,t}\|^2}{2}\right)}. \tag{10}$$

Further sorting, thus, we derive Eq. (1), completing the proof.

7.2. Theorem 2

Theorem 2. *Given the denoising process modeled by a score-based probability ratio function $s_\theta(z^g_{n,t})$, defined as $s_\theta = \frac{p_{data}(z^g_{n,0})}{p_{data}(z^g_{n,t})}$, this paper defines a learnable approximation using a parameterized score function f_θ, such that the probability ratio can be estimated as:*

$$s_\theta(z^g_{n,t}) = \frac{\exp\left(f_\theta(z^g_{n,t}, z^g_{n,0})\right)}{\sum_{y \in z^g_{n,0:t-1}} \exp\left(f_\theta(z^g_{n,t}, y)\right)}, \tag{11}$$

Proof. To derive Eq. (11), we start from the definition of the score-based probability ratio:

$$s_\theta(z^g_{n,t}) = \frac{p_\theta(z^g_{n,0})}{p_\theta(z^g_{n,t})}. \tag{12}$$

Using Bayes' theorem, we can express the conditional probability as:

$$p_\theta(z^g_{n,0}|z^g_{n,t}) = \frac{p(z^g_{n,t}|z^g_{n,0}) p_\theta(z^g_{n,0})}{p(z^g_{n,t})}. \tag{13}$$

Taking the logarithm on both sides, we define a learnable function $f_\theta(z^g_{n,t}, z^g_{n,0})$ that approximates:

$$f_\theta(z^g_{n,t}, z^g_{n,0}) \approx \log p_\theta(z^g_{n,0}|z^g_{n,t}). \tag{14}$$

Given the forward diffusion process follows:

$$p(z^g_{n,t}|z^g_{n,0}) = \mathcal{N}(z^g_{n,t}; \sqrt{\bar{\alpha}^g_t} z^g_{n,0}, (1-\bar{\alpha}^g_t) I), \tag{15}$$

and the marginal distribution:

$$q(z^g_{n,t}) \approx \mathcal{N}(z^g_{n,t}; 0, I), \tag{16}$$

we obtain:

$$f_\theta(z^g_{n,t}, z^g_{n,0}) = -\frac{\|z^g_{n,t} - \sqrt{\bar{\alpha}^g_t} z^g_{n,0}\|^2}{2(1-\bar{\alpha}^g_t)} + \frac{\|z^g_{n,t}\|^2}{2}. \quad (17)$$

To ensure numerical stability and gradient optimization, we normalize $s_\theta(z^g_{n,t})$ using softmax over the set of possible denoising states:

$$s_\theta(z^g_{n,t}) = \frac{\exp\left(f_\theta(z^g_{n,t}, z^g_{n,0})\right)}{\sum_{y \in z^g_{n,0:t-1}} \exp\left(f_\theta(z^g_{n,t}, y)\right)}. \quad (18)$$

Thus, we have derived Eq. (11), which provides a parameterized score function for probability ratio estimation.

7.3. Theorem 3

Theorem 3. *To achieve the optimal score entropy [50] which is demonstrated on Eq. (21), the selection step choose j items where each $z^g_{n,t}$ satisfies $se = 0$, i.e.,*

$$s_\theta(z^g_{n,t}) \approx \frac{p_{data}(y)}{p_{data}(z^g_{n,t})} \quad (19)$$

Proof. To prove the Theorem 3, we divide this proof into three parts: The first is to determine **the optimization target of the model approximation**. The second is to determine **the iterative process of the model optimization target**. The third is to prove **the convergence validity of the iterative process**.

1) **The optimization target of the model approximation**

According to the denoising score entropy proposed by Lou *et al*. [50], the Mamba block loss function can be defined as follows:

$$L_{se} = \mathbb{E}_{z^g_{n,0} \sim p_0, z^g_n \sim p(\cdot|z^g_{n,0})} se \quad (20)$$

To minimize the loss function, the se should be closed to value 0. And based on the score entropy loss [50], the se can be described as:

$$se = \sum_{y \in z^g_{n,0:t-1}} \omega^g_{z^g_{n,t}} \left(s_\theta(z^g_{n,t}) - \frac{p_{data}(y)}{p_{data}(z^g_{n,t})} \log s_\theta(z^g_{n,t}) + K\left(\frac{p_{data}(y)}{p_{data}(z^g_{n,t})}\right)\right), \quad (21)$$

where $K(a) = a(\log a - 1)$ a normalization term that ensures the loss is non-negative. And weights $\omega^g_{z^g_{n,t}} \in (0, 1)$ can adjust the weights assigned to different noise latent representations. This can improve optimization efficiency by explicitly selecting important point pairs. For example, higher weights can be assigned to noise latent representations that may introduce larger errors within a specific range, thereby guiding the update of the model. And ultimately control the final total se to be close to 0. And $s_\theta(z^g_{n,t})$ is n-th noise latent representation of the model predicted score ratio at t-th denoising step.

To determine the necessary conditions for minimizing se, we compute the partial derivative with respect to $s_\theta(z^g_{n,t})$:

$$\frac{\partial se}{\partial s_\theta(z^g_{n,t})} = \sum_{y \in z^g_{n,0:t-1}} \omega^g_{z^g_{n,t}} \left(1 - \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})} \frac{1}{s_\theta(z^g_{n,t})}\right). \quad (22)$$

Setting the gradient to zero for optimization,

$$1 - \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})} \frac{1}{s_\theta(z^g_{n,t})} = 0. \quad (23)$$

Rearranging the terms, we obtain:

$$s_\theta(z^g_{n,t}) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})}. \quad (24)$$

Thus, at the optimal solution, the predicted score function must exactly match the empirical probability ratio.

For model parameters θ, we analyze the gradient:

$$\frac{\partial se}{\partial \theta} = \sum_{y \in z^g_{n,0:t-1}} \omega^g_{z^g_{n,t}} \left(\frac{\partial s_\theta(z^g_{n,t})}{\partial \theta} - \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})} \frac{1}{s_\theta(z^g_{n,t})} \frac{\partial s_\theta(z^g_{n,t})}{\partial \theta}\right). \quad (25)$$

For gradient convergence, we set the derivative to zero:

$$\frac{\partial s_\theta(z^g_{n,t})}{\partial \theta} \left(1 - \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})} \frac{1}{s_\theta(z^g_{n,t})}\right) = 0. \quad (26)$$

Since the gradient term $\frac{\partial s_\theta(z^g_{n,t})}{\partial \theta}$ is nonzero for model updates, the following condition must hold:

$$1 - \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})} \frac{1}{s_\theta(z^g_{n,t})} = 0, \quad (27)$$

which again yields the optimal condition:

$$s_\theta(z^g_{n,t}) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})}. \quad (28)$$

In summary, the necessary conditions for minimizing the Score Entropy Loss and ensuring the optimal score function are:

- The predicted score function must satisfy:

$$s_\theta(z^g_{n,t}) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})}. \quad (29)$$

- The gradient with respect to the model parameters must satisfy:

$$\frac{\partial s_\theta(z^g_{n,t})}{\partial \theta}\left(1-\frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})}\frac{1}{s_\theta(z^g_{n,t})}\right)=0. \tag{30}$$

These conditions imply that when the model learns the correct probability ratio, the gradient becomes zero, leading to optimal convergence of the Score Entropy Loss. Therefore, optimizing $s_\theta(z^g_{n,t})$ to match $\frac{p_{\text{data}}(y)}{p_{\text{data}}(z^g_{n,t})}$ is both a necessary and sufficient condition for achieving the lowest possible loss.

Based on Eq. (26), $\theta = \{H^g_{n,t}, A, B, C, D, \Delta\}$ represent the state space in the block. We can obtain the selected noise latent representation $z^g_{n,t}$ by updating the computation in the state space architecture from Mamba-2 [26], which can be defined as follows:

$$H^g_{n,t} = \bar{A}H^g_{n,t-1} + \bar{B}z^g_{n,t} \tag{31}$$

$$z^g_{n-1,t} = CH^g_{n,t} + Dz^g_{n,t} \tag{32}$$

$$\bar{A} = \exp(\Delta A) \tag{33}$$

$$\bar{B} = (\Delta A)^{-1}\cdot(\exp(\Delta A) - I)\cdot \Delta B \tag{34}$$

where $H^g_{n,t}$ represents the hidden state representation, A and B control the evolution of hidden states and latent space noise vector inputs, respectively, C governs the hidden state representation of the target output and D manages the nonlinear skip connection for latent space noise vector inputs. Δ denotes the learnable time parameter.

2) **The iterative process of the model optimization target** Considering the parameters in θ, they are updated by the following steps. First, **the update of A and $\bar{A}$**. Given that $\bar{A}$ controls the recursive evolution of hidden state $H^g_{n,t}$ based on A and Δ, we can gain the relationship in Eq. (33). So, the gradient can be described as follows:

$$\frac{\partial \mathcal{L}}{\partial A} = \frac{\partial \mathcal{L}}{\partial \bar{A}}\cdot\frac{\partial \bar{A}}{\partial A} \tag{35}$$

where

$$\frac{\partial \bar{A}}{\partial A} = \Delta\cdot\exp(\Delta A) \tag{36}$$

then through backpropagation to calculate the gradient of $\mathcal{L}$ to $\bar{A}$ and combined with the chain rule to update A.

Second, **the update of B and $\bar{B}$**. Given that the definition of $\bar{B}$ in Eq. (34), the gradient can be described as follows (familiar with the update rule of A):

$$\frac{\partial \mathcal{L}}{\partial B} = \frac{\partial \mathcal{L}}{\partial \bar{B}}\cdot\frac{\partial \bar{B}}{\partial B} \tag{37}$$

where gradient transfer involves matrix derivation, which requires considering the derivative rule of matrix multiplication. Finally, the chain rule depends on the gradients of ΔA and ΔB.

Third, **the update of C**. Given that C controls the hidden state and its direct contribution to the output $z^g_{n-1,t}$ is as Eq. (32) defined, the gradient can be described as follows:

$$\frac{\partial \mathcal{L}}{\partial C} = \frac{\partial \mathcal{L}}{\partial z^g_{n-1,t}}\cdot\frac{\partial z^g_{n-1,t}}{\partial C} \tag{38}$$

where

$$\frac{\partial z^g_{n-1,t}}{\partial C} = H^g_{n,t} \tag{39}$$

So the update rule can be described as follows:

$$C \leftarrow C - \eta\frac{\partial \mathcal{L}}{\partial C} \tag{40}$$

where η is the learning rate.

Fourth, **the update of D**. Given that D governs the skip connection and directly act on $z^g_{n,t}$, the gradient can be defined as follows:

$$\frac{\partial \mathcal{L}}{\partial D} = \frac{\partial \mathcal{L}}{\partial z^g_{n-1,t}}\cdot\frac{\partial z^g_{n-1,t}}{\partial D} \tag{41}$$

where

$$\frac{\partial z^g_{n-1,t}}{\partial D} = z^g_{n,t} \tag{42}$$

Fifth, **the update of Δ**. Δ denotes the learnable time parameter and affects the dynamic behavior of $\bar{A}$ and $\bar{B}$. So the gradient can be defined as follows:

$$\frac{\partial \mathcal{L}}{\partial \Delta} = \frac{\partial \mathcal{L}}{\partial \bar{A}}\cdot\frac{\partial \bar{A}}{\partial \Delta} + \frac{\partial \mathcal{L}}{\partial \bar{B}}\cdot\frac{\partial \bar{B}}{\partial \Delta} \tag{43}$$

where

$$\frac{\partial \bar{A}}{\partial \Delta} = A\cdot\exp(\Delta A) \tag{44}$$

$$\begin{aligned} f(A,B,\Delta) = &-(\Delta A)^{-1}A(\Delta A)^{-1}(\exp(\Delta A)-I)\Delta B \\ &+(\Delta A)^{-1}(A\exp(\Delta A))\Delta B \\ &+(\Delta A)^{-1}(\exp(\Delta A)-I)B \end{aligned} \tag{45}$$

In this problem, the structure of the state space model and the diffusion model provide theoretical support for the strong convexity of the loss function and the Lipschitz property of the gradient. First, the stability of the state space model leads to the hidden state update equation:

$$H^g_{n,t} = \bar{A}H^g_{n,t-1} + \bar{B}z^g_{n,t} \tag{46}$$

where $\bar{A} = \exp(\Delta A), \bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I)\Delta B$ is generated via matrix exponential. It has the following characteristics:

- If A is a stable matrix (all eigenvalues have negative real parts), then the modulus of the eigenvalues of $\bar{A}$ is less than 1, which ensures that the hidden state does not diverge.
- The state update equation is linear, so the gradient of the parameters A, B, C, D is linearly solvable, making it easy to optimize.

Secondly, given the characteristics of the diffusion model, there is a score ratio prediction loss function:

$$\mathcal{L} = \mathbb{E}_{z^g_{n,t},p}\left[\| \frac{p_{data}(y)}{p_{data}(z^g_{n,t})} - s_\theta(z^g_{n,t})\|_2^2\right] \tag{47}$$

where $\mathcal{L}$ is in squared error form and is therefore a convex function (subconvexity). Then the gradient can be expressed as follows:

$$\nabla_\theta \mathcal{L} = 2\mathbb{E}_{z^g_{n,t},p}\left[\|(\frac{p_{data}(y)}{p_{data}(z^g_{n,t})} - s_\theta(z^g_{n,t}))\nabla_\theta s_\theta(z^g_{n,t})\|_2^2\right] \tag{48}$$

where the gradient is a linear combination of θ and satisfies the Lipschitz continuity condition.

To sum up, combined with the model parameters $\theta = A, B, C, D, \Delta$, there is the following convergence of the specific parameter updating process.

First for the hidden state update:

$$H^g_{n,t} = \bar{A}H^g_{n,t-1} + \bar{B}z^g_{n,t} \tag{49}$$

where A is a stable matrix, $\bar{A}$ is stable, ensuring that the hidden state does not diverge.

Second for output calculation:

$$z^g_{n-1,t} = CH^g_{n,t} + Dz^g_{n,t} \tag{50}$$

and it is a linear transformation, which ensures the stability of the gradient solution for C and D.

Third for time step parameters Δ, it is a learnable parameter of the time scale, which is directly related to the discretization in the state space model. It is updated by the chain rule as follows:

$$\frac{\partial \mathcal{L}}{\partial \Delta} = \frac{\partial \mathcal{L}}{\partial \bar{A}} \cdot \frac{\partial \bar{A}}{\partial \Delta} + \frac{\partial \mathcal{L}}{\partial \bar{B}} \cdot \frac{\partial \bar{B}}{\partial \Delta} \tag{51}$$

among this, in the discretization formula, $\bar{A}$ and $\bar{B}$ are exponential functions with continuous and differentiable gradients, which are easy to converge.

3) **The convergence validity of the iterative process**

In order to ensure the convergence of the above iterative process, the following conditions usually need to be met:

- The convergent objective function $\mathcal{L}$ is a continuously differentiable function with respect to parameter θ and it is strongly convex or subconvex (at least a convex function).
- Make sure the learning rate satisfies $0 < \eta < 2/L$ where L is the Lipschitz constant for the gradient $\nabla_\theta \mathcal{L}$ of the convergent objective function (the upper bound on the rate of change of the gradient).
- The matrix $\bar{A}$ (generated by discretization) is stable, that is, the magnitude of its eigenvalues is less than 1.

When the above convergence conditions are met, assuming that the convergence target $\mathcal{L}$ function is a μ-strongly convex function (strong convexity is a stricter form of convex function), the convergence of gradient descent can be proved by the following formula. First, the updated formula for gradient descent is given:

$$\theta^{k+1} = \theta^k - \eta \nabla_\theta \mathcal{L}(\theta^k) \tag{52}$$

where θ^k is the parameter vector at the k-th iteration.

Secondly, the properties of strongly convex functions are given, that is, if the convergent objective function $\mathcal{L}$ is μ-strongly convex and the Lipschitz constant of the gradient is L, then the error of the gradient descent method will converge at an exponential rate:

$$\mathcal{L}(\theta^k) - \mathcal{L}(\theta^*) \leq \rho^k \big(\mathcal{L}(\theta^0) - \mathcal{L}(\theta^*)\big) \tag{53}$$

where $\rho = 1 - 2\eta\mu$ is the convergence rate ($0 < \rho < 1$), and θ^* is the global optimum.

Third, if the Lipschitz gradient condition is satisfied, that is, $\nabla_\theta \mathcal{L}$ is L-Lipschitz continuous:

$$\|\nabla_\theta \mathcal{L}(\theta_1) - \nabla_\theta \mathcal{L}(\theta_2)\| \leq L\|\theta_1 - \theta_2\| \tag{54}$$

then selecting a learning rate $0 < \eta < \frac{2}{L}$ ensures convergence.

Algorithm 1 Gradient Descent Algorithm

Input: Initialize parameters A, B, C, D, and Δ.
repeat
Calculate the loss $\mathcal{L}$.
Compute the gradient of $\mathcal{L}$ with respect to A, B, C, D, and Δ using the chain rule.
Update each parameter using the gradient descent rule.
Perform backpropagation to compute: $\nabla_\theta \| \frac{p_{data}(y)}{p_{data}(z^g_{n,t})} - s_\theta(z^g_{n,t})\|_2^2$.
until convergence

In general, the process of update and convergence can be summarized in Algorithm 1. Through repeated iterations, the model parameter θ will be gradually optimized, so that the convergence objective function $\mathcal{L}$ will be converged and se gradually approaches 0, that is, s_θ approaches $\frac{p_{data}(y)}{p_{data}(z^g_{n,t})}$. Then j items of noise latent representation $z^g_{n,t}$ that satisfy all the above conditions will be selected, and

the model will proceed to the next step of denoising in the direction of these j items.

Above all, in the inference stage, the model will choose the best noise latent representation of image patch or text embedding, including j items to restore the image or text. Due to this, the model has already learned from the datasets that should be focused on and ignored. Compared with the Transformer models, which need to calculate all image patches or text embeddings, it will shorten the inference time when generating high-resolution images or long-sequence text. The results are shown in the main paper Section 5.3.1 Performance Analysis.

8. Appendix B

8.1. Denoising process based on DPM-Solver

Based on the diffusion denoising model trained by Score Entropy Loss, we hope to combine DPM-Solver (Diffusion Probabilistic Model Solver)[51] in the inference stage to reduce sampling steps and improve inference efficiency.

DPM-Solver is a high-order ODE-solving method for diffusion models. It constructs partial differential equations (ODEs) and uses numerical solution techniques to accelerate the diffusion denoising process. It can restore high-quality data from Gaussian noise in a minimal number of steps (such as 10 steps) without sacrificing model performance.

The core idea of DPM-Solver is to reformulate the inverse diffusion process of the diffusion model as an ordinary differential equation (ODE) and solve it efficiently using numerical methods. For the standard diffusion model, we have:

$$\frac{dz^g_{n,t}}{dt} = -\frac{1}{2}\beta_t z^g_{n,t} + \sqrt{\beta_t}\epsilon^g_{n,t}, \quad \epsilon^g_{n,t} \sim \mathcal{N}(0, I). \tag{55}$$

DPM-Solver estimates $\epsilon_\theta(z^g_{n,t}, t)$ by denoising the score matching, which can be rewritten as:

$$\frac{dz^g_{n,t}}{dt} = f_\theta(z^g_{n,t}, t), \tag{56}$$

where the formula describes the rate of change of the latent variable $z^g_{n,t}$ in the time t dimension, and its evolution process can be accelerated by numerical solution methods.

In the Mamba decoder trained with Score Entropy Loss, we learn:

$$s_\theta(z^g_{n,t}) = \frac{\exp\left(f_\theta(z^g_{n,t}, z^g_{n,0})\right)}{\sum_{y \in z^g_{n,0:t-1}} \exp\left(f_\theta(z^g_{n,t}, y)\right)}. \tag{57}$$

Therefore, in the DPM-Solver framework, we hope to use this ratio's gradient information to directly construct the ODE and reduce the number of sampling steps during inference.

First, we need to compute denoised ODE. DPM-Solver uses Score Matching technology [51] to predict the noise $\epsilon_\theta(z^g_{n,t}, t)$ through a neural network, and then calculates it according to the denoising ODE:

$$\frac{dz^g_{n,t}}{dt} = -\frac{1}{2}\beta_t \left(\frac{z^g_{n,t} - \sqrt{\bar{\alpha}^g_t} z^g_{n,0}}{1 - \bar{\alpha}^g_t} \right), \tag{58}$$

furthermore, we can calculate based on Score Entropy [50]:

$$\frac{dz^g_{n,t}}{dt} = -\frac{1}{2}\beta_t s_\theta(z^g_{n,t}) \nabla_z \log p_\theta(z^g_{n,0} | z^g_{n,t}), \tag{59}$$

where $\nabla_z \log p_\theta(z^g_{n,0} | z^g_{n,t})$ is calculated by se, $s_\theta(z^g_{n,t}$ is predicted probability ratios through neural networks. This formula describes the ODE trajectory from the noisy state $z^g_{n,t}$ to the denoised state $z^g_{n,0}$.

We then use DPM-Solver to perform inference. For the first-order approximation method, the basic form of DPM-Solver is the first-order ODE approximation:

$$z^g_{n,t} \approx z^g_{n,t-\Delta t} - \frac{1}{2}\beta_t \left(\frac{z^g_{n,t} - \sqrt{\bar{\alpha}^g_t} z^g_{n,0}}{1 - \bar{\alpha}^g_t} \right) \Delta t, \tag{60}$$

by using $s_\theta(z^g_{n,t})$ calculated by Score Entropy Loss, we can further rewrite the formula:

$$z^g_{n,t} \approx z^g_{n,t-\Delta t} - \frac{1}{2}\beta_t s_\theta(z^g_{n,t}) \nabla_z \log p_\theta(z^g_{n,0} | z^g_{n,t}) \Delta t. \tag{61}$$

The formula can be directly used to update the denoising process to achieve efficient sampling iteratively.

Furthermore, DPM-Solver uses second-order numerical methods [51] to improve accuracy:

$$z^g_{n,t} = z^g_{n,t-\Delta t} + \frac{\Delta t}{2}\left[f_\theta(z^g_{n,t}, t) + f_\theta(z^g_{n,t-\Delta t}, t - \Delta t) \right] \tag{62}$$

which allows us to complete denoising inference in a very small number of iterations (e.g., 10-20 steps), significantly speeding up the computation compared to normal diffusion sampling (e.g., 1000 steps).

Algorithm 2 Mamba-Based Inference with DPM-Solver

Input: Noisy latent state $z^g_{n,t}$.
repeat
Predict the score function $s_\theta(z^g_{n,t})$ for computing the denoising ODE.
Apply DPM-Solver update rule: $z^g_{n,t} \leftarrow z^g_{n,t-\Delta t} + \frac{\Delta t}{2}\left[f_\theta(z^g_{n,t}, t) + f_\theta(z^g_{n,t-\Delta t}, t - \Delta t) \right]$.
until gain the $z^g_{n,t}$

9. Appendix C

9.1. Model Configuration

Configuration	Value
Size	7B
Mamba block	49
Hidden Dimension	2048
GFlops	424
Optimizer	AdamW
Learning Rate	0.0001
Weight Decay	-
Training Epochs	1
Sampling step	500000
EMA	0.9999
Patch size	2×2
Maximum Token Length	512

Table 1. Parameter settings for MDM.

10. Appendix D

10.1. SentencePiece (Unigram BPE)

SentencePiece (Unigram BPE) [45] provides an optimal subword-based tokenization approach that enables improved generalization and adaptability for handling both textual and multimodal data.

10.1.1. Theoretical Background

SentencePiece employs a probabilistic model based on a Unigram Language Model (ULM), where each sentence x is decomposed into a sequence of subwords s_i with a likelihood function:

$$p(x) = \prod_i p(s_i), \tag{63}$$

where each subword unit s_i is assigned a probability estimated from training data. Unlike traditional Byte-Pair Encoding (BPE), which deterministically merges frequent subword pairs, the Unigram BPE method probabilistically learns an optimal vocabulary while gradually discarding subwords with lower contributions.

To train SentencePiece, an initial vocabulary is constructed using all possible subword combinations, after which an iterative Expectation-Maximization (EM) optimization is performed. At each iteration, subwords contributing the least to sequence likelihoods are removed, leading to an optimal vocabulary.

10.1.2. Training Procedure

The training of the SentencePiece model is conducted on a large-scale dataset containing both pure-text corpora and multimodal text-image descriptions. Given the multimodal nature of our dataset, we mix textual data from Ultrachat and text descriptions from JourneyDB and ImageNet to ensure cross-modal adaptability.

Dataset Preprocessing: To prepare the dataset, raw text is extracted, normalized, and formatted as a line-separated corpus file. The dataset mixing strategy follows:

- Extract textual information from Ultrachat.
- Concatenate textual descriptions from JourneyDB and ImageNet.
- Remove redundant, low-quality, or excessively short text samples.
- Shuffle the corpus to prevent dataset bias.

SentencePiece Model Training: The SentencePiece Unigram BPE model is trained using the following configuration:

```
import sentencepiece as spm
spm.SentencePieceTrainer.train(
    input="text_data.txt",
    # Training corpus
    model_prefix="unigram_bpe",
    # Output model prefix
    vocab_size=32000,
    # Vocabulary size
    model_type="unigram",
    # Unigram-based BPE
    character_coverage=0.9995,
    # Coverage for rare characters
    num_threads=8,
    # Parallel training
    input_sentence_size=1000000,
    # Sample size
    shuffle_input_sentence=True
    # Shuffle corpus
)
```

This results in two key output files: `unigram_bpe.model` (binary model for tokenization) and `unigram_bpe.vocab` (vocabulary list with probabilities).

10.1.3. Evaluation and Optimization Strategies

The effectiveness of the trained tokenization model is evaluated based on tokenization efficiency and generalization capability. The following criteria are considered:

- **Subword Granularity**: The trade-off between word and character-level tokenization.
- **Out-of-Vocabulary (OOV) Rate**: The ability to handle unseen words.
- **Multimodal Alignment**: The compatibility of subword embeddings with image features in the latent space.

Given the computational constraints of multimodal diffusion models, we optimize the SentencePiece model with:

- Selecting an optimal `vocab_size` ($16K$-$32K$) to balance representation and sequence length.
- Applying dataset mixture strategies to enhance generalization across different data distributions.
- Ensuring tokenization stability by enforcing `character_coverage` 0.9995 to capture rare textual variations.

11. Appendix E

11.1. Complexity

Since the size of the noisy latent encoder (VAE) is significantly smaller than that of the diffusion decoder (Mamba), we will focus our analysis on the computational complexity of the diffusion decoder. According to [61], the complexity of each Mamba block is $\mathcal{O}(LN^2)$, where L is the length of the input data and N refers to the size of each parameter ($\{H^g_{n,t}, A, B, C, D, \Delta\}$) in the state space. The diffusion decoder is composed of M Mamba blocks, resulting in an overall computational complexity of $\mathcal{O}(MLN^2)$.

For comparison, consider an equivalent end-to-end transformer model optimized with GQA [1, 71, 89]. This model maintains the same input length L and GQA module dimension N. With M layers and a grouping parameter G, its computational complexity is $\mathcal{O}(ML^2N/G)$.

Determining which complexity is superior between $\mathcal{O}(MLN^2)$ and $\mathcal{O}(ML^2N/G)$ can be challenging. However, it is important to note that N can be significantly smaller than L/G when L is very large. As a result, the proposed MDM can achieve greater computational efficiency than end-to-end transformer models when processing high-resolution images and long-sequence texts.

12. Appendix F

12.1. Image generation

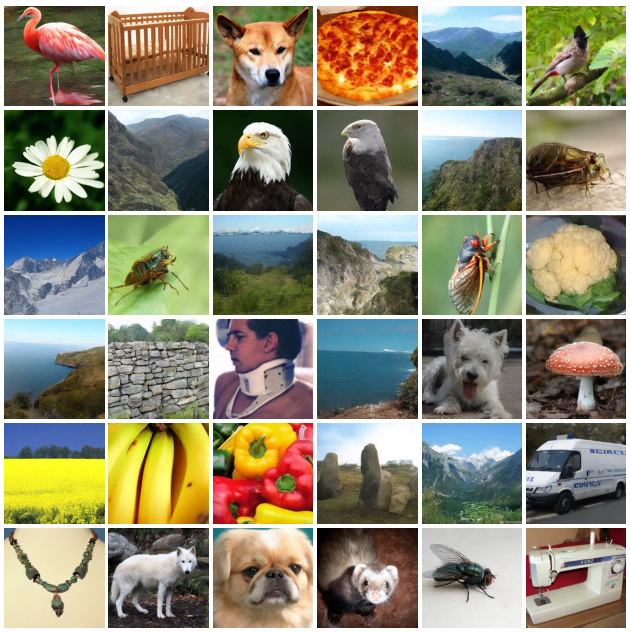

Figure 1. Image generation with CFG on ImageNet [13] 256 × 256.

12.2. Image generation on COCO and Flickr

Figure 2. Image generation on COCO [40] caption text.

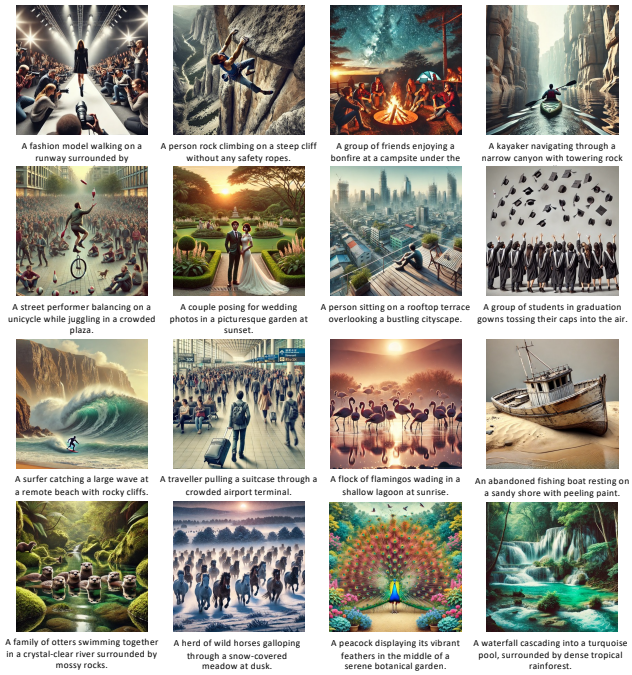

Figure 3. Image generation on Flickr 30K [84] caption text.

13. Appendix G

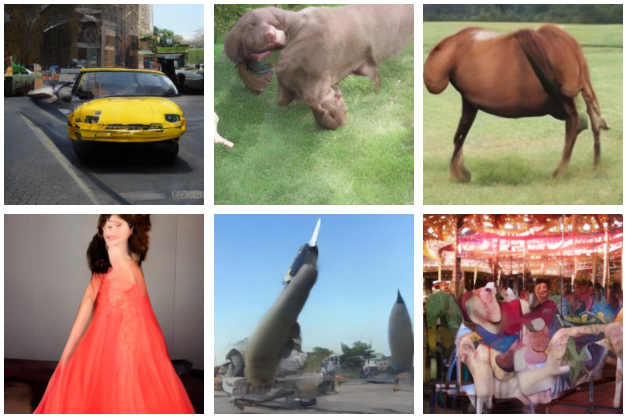

Figure 4. Drawbacks in image generation.

13.1. Drawbacks

While MDM demonstrates strong performance across various tasks and enhanced processing speed for high-resolution images and long text sequences (as shown in the main paper Section 5.3.1 Performance Analysis), it faces several limitations. The model shows reduced efficiency when handling low-resolution images or short text sequences, and its overall performance still trails behind traditional multi-modal pre-trained models. Furthermore, the model exhibits hallucination issues. These limitations represent key areas for future improvement.

It can be observed from Fig. 4 that MDM still generates a small number of defective images, such as image deformation, collapse, distortion, and blurring. This may be due to the model's scale being insufficient and limitations in how each modality's data is represented in the decoder. Additionally, the diffusion reduction process might experience some instability, which could lead to subpar sampling results. Therefore, there is still potential for further improvements to the model to address these issues.

The partial performance results of the model on the Flickr 30K dataset reveal significant challenges, particularly when dealing with complex text data that requires generating intricate images, especially those involving people and animals. The model often loses important details, such as facial features and the depiction of limbs. Additionally, it exhibits a tendency to be inefficient and make errors, such as repetitively copying and pasting certain objects, resulting in a dilution of detail for those entities and the generation of instances that do not accurately match the accompanying descriptive language (as shown in Figs. 3 and 5). The main reason for the above problems is that the Flickr 30K dataset emphasizes the correlation between different modal semantics rather than focusing solely on classification or recognition tasks like the COCO dataset. This means that the model needs stronger capabilities for multi-modal semantic understanding. The MDM model employs a unified modal fusion decoder under a constrained scale, which may limit its ability to enhance semantic understanding compared to traditional models. Therefore, the MDM model needs continuous optimization.

Figure 5. Drawbacks in generating complex captions images.