Boarder Impact

Mirror Diffusion Models (MDMs) advance the recent development of diffusion models to complex domains subjected to convex constrained sets. This opens up new possibilities for MDMs to serve as preferred models for generating samples that live in *e.g.,* simplices and balls. Additionally, MDMs introduce an innovative application of constrained sets as a watermarking technique. This has the potential to address concerns related to unethical usage and safeguard the copyright of generative models. By incorporating constrained sets into the generating process, MDMs offer a means to prevent unauthorized usage and ensure the integrity of generated content.

A Derivation of Mirror Mappings

Here, we provide addition derivation of $\nabla\phi^*$. Computation of $\nabla\phi(x)$ and $\nabla^2\phi^*(y)$ follow straightforwardly by differentiating $\phi(x)$ and $\nabla\phi^*(y)$ w.r.t. x and y, respectively.

ℓ_2-Ball Since the gradient map also reverses the mirror map, we aim to rewrite $y = \frac{2\gamma}{R-\|x\|_2^2}x$ as $x = f(y) = \nabla\phi^*_{\text{ball}}(y)$. Solving the second-order polynomial,

$$\|y\|_2^2 = \left(\frac{2\gamma}{R-\|x\|_2^2}\right)^2\|x\|_2^2, \tag{19}$$

yields

$$\|x\|_2^2 = R + \frac{2\gamma}{\|y\|_2^2}\left(\gamma - \sqrt{R\|y\|_2^2+\gamma^2}\right). \tag{20}$$

With that, we can rewrite Equation (11) by

$$x = \frac{R-\|x\|_2^2}{2\gamma}y \overset{(20)}{=} \frac{\sqrt{R\|y\|_2^2+\gamma^2}-\gamma}{\|y\|_2^2}y = \frac{R}{\sqrt{R\|y\|_2^2+\gamma^2}+\gamma}y.$$

Simplex Standard calculations in convex analysis [28] shows

$$\phi^*_{\text{simplex}}(y) = \log\left(1+\sum_i^d e^{y_i}\right). \tag{21}$$

Differentiating Equation (21) w.r.t. y yields $\nabla\phi^*_{\text{simplex}}$ in Equation (13).

Polytope Since the gradient map also reverses the mirror map, we aim to inverse

$$y = \sum_{i=1}^m s_i(\langle a_i, x\rangle)a_i + \sum_{j=m+1}^d \langle a_j, x\rangle a_j. \tag{22}$$

When all d constraints are orthonormal, taking inner product between y and each a yields

$$\langle a_i, y\rangle = s_i(\langle a_i, x\rangle), \qquad \langle a_j, y\rangle = \langle a_j, x\rangle. \tag{23}$$

Therefore, we can reconstruct x from y via

$$\begin{aligned} x &= \sum_{i=1}^m \langle a_i, x\rangle a_i + \sum_{j=m+1}^d \langle a_j, x\rangle a_j \\ &\overset{(23)}{=} \sum_{i=1}^m s_i^{-1}(\langle a_i, y\rangle)a_i + \sum_{j=m+1}^d \langle a_j, y\rangle a_j, \end{aligned}$$

which defines $x = \nabla\phi^*_{\text{polytope}}(y)$. For completeness, the Hessian can be presented compactly as

$$\nabla^2\phi^*_{\text{polytope}}(y) = \boldsymbol{I} + \boldsymbol{A}\Sigma\boldsymbol{A}^\top, \tag{24}$$

where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{A} := [a_1, \cdots, a_m]$ is a d-by-m matrix whose column vector a_i corresponds to each constraint, and $\Sigma \in \mathbb{R}^{m\times m}$ is a diagonal matrix with leading entries

$$[\Sigma]_{ii} = \frac{\partial s_i^{-1}(z)}{\partial z}|_{z=\langle a_i,y\rangle} - 1 \overset{(18)}{=} \frac{b_i-c_i}{2}\left(1-\tanh^2(\langle a_i, y\rangle)\right) - 1.$$

B Additional Remarks on Polytope

Derivation of Equation (17) Since the subspaces spanned by $\{a_i\}$ and $\{a_j\}$ are orthogonal to each other, we can rewrite (15) as

$$\nabla\phi_{\text{polytope}}(x) = \sum_{i=1}^{m} s_i(\langle a_i, x\rangle)a_i + \left(x - \sum_{i=1}^{m}\langle a_i, x\rangle a_i\right) = x + \sum_{i=1}^{m}\left(s_i(\langle a_i, x\rangle) - \langle a_i, x\rangle\right)a_i.$$

$\nabla\phi^*_{\text{polytope}}(y)$ follows similar derivation.

Generalization to non-orthonormal constraints The mirror maps of a polytope, as described in Equations (15) to (17), can be seen as operations that manipulate the coefficients associated with the bases defined by the constraints. This understanding allows us to extend the computation to non-orthonormal constraints by identifying the corresponding "coefficients" through a change of bases, utilizing the reproducing formula:

$$x = \sum_{i=1}^{d}\langle \tilde{a}_i, x\rangle a_i, \text{ where } \tilde{a}_i \text{ is the } i\text{-th row of } (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top,$$

and $\boldsymbol{A} := [a_1, \cdots, a_m]$. Similarly, we have $y = \sum_{i=1}^{d}\langle \tilde{a}_i, y\rangle a_i$. Applying similar derivation leads to

$$\nabla\phi_{\text{poly}}(x) = x + \sum_{i=1}^{m}\left(s_i(\langle \tilde{a}_i, x\rangle) - \langle \tilde{a}_i, x\rangle\right)a_i, \ \nabla\phi^*_{\text{poly}}(y) = y + \sum_{i=1}^{m}\left(s_i^{-1}(\langle \tilde{a}_i, y\rangle) - \langle \tilde{a}_i, y\rangle\right)a_i.$$

C Experiment Details & Additional Results

Table 7: The concentration parameter α of each Dirichlet distribution in simplices constrained sets.

d	3	3	7	9	20
α	$[2, 4, 8]$	$[1, 0.1, 5]$	$[1, 2, 2, 4, 4, 8, 8]$	$[1, 0.5, 2, 0.3, 0.6, 4, 8, 8, 2]$	$[0.2, 0.4, \cdots, 4, 4.2]$

Table 8: Hyperparameters of the polytope $\mathcal{M} := \{x \in \mathbb{R}^d : c_i < a_i^\top x < b_i\}$ for each dataset and watermark precision. Note that we fix $b = b_i = -c_i$ in practice.

	FFHQ 64×64 (uncon)			AFHQv2 64×64 (uncon)		
Precision	*59.3%*	*71.8%*	*93.3%*	*56.9%*	*75.0%*	*92.7%*
Number of constraints m	7	20	100	4	20	100
Constraint range b	1.05	1.05	1.05	0.9	0.9	0.9

Dataset & constrained sets

- *ℓ_2-balls constrained sets*: For $d = 2$, we consider the Gaussian Mixture Model (with variance 0.05) and the Spiral shown respectively in Figures 1 and 3. For $d = \{6, 8, 20\}$, we place d isotropic Gaussians, each with variance 0.05, at the corner of each dimension, and reject samples outside the constrained sets.
- *Simplices constrained sets*: We consider Dirichlet distributions [48], $\text{Dir}(\alpha)$, with various concentration parameters α detailed in Table 7.
- *Hypercube constrained sets*: For all dimensions $d = \{2, 3, 6, 8, 20\}$, we place d isotropic Gaussians, each with variance 0.2, at the corner of each dimension, and either reject ($d = \{2, 3, 6, 8\}$) or reflect ($d = 20$) samples outside the constrained sets.
- *Watermarked datasets and polytope constrained sets*: We follow the same data preprocessing from EDM[3] [38] and rescale both FFHQ and AFHQv2 to 64×64 image resolution. For the polytop constrained sets $\mathcal{M} := \{x \in \mathbb{R}^d : c_i < a_i^\top x < b_i, \forall i\}$, we construct a_i from orthonormalized Gaussian random vectors and detail other hyperparameters in Table 8.

[3]https://github.com/NVlabs/edm, released under Nvidia Source Code License.

Implementation All methods are implemented in PyTorch [72]. We adopt ADM[4] and EDM[3] [38] respectively as the MDM's diffusion backbones for constrained and watermarked generation. We implemented Reflected Diffusion [18] by ourselves as their codes have not yet been made available, and used the official implementation[5] of Reflected Diffusion [25] in Table 11. We also implemented Simplex Diffusion [60], but as observed in previous works [25], it encountered computational instability especially when computing the modified Bessel functions.

Training For constrained generation, all methods are trained with AdamW [73] and an exponential moving average with the decay rate of 0.99. As standard practices, we decay the learning rate by the decay rate 0.99 every 1000 steps. For watermarked generation, we follow the default hyperparameters from EDM[3] [38]. All experiments are conducted on two TITAN RTXs and one RTX 2080.

Network For constrained generation, all networks take (y, t) as inputs and follow

$$\texttt{out} = \texttt{out\_mod(norm(y\_mod(}\ y\ \texttt{)} + \texttt{t\_mod(timestep\_embedding(}\ t\ \texttt{))))},$$

where `timestep_embedding(·)` is the standard sinusoidal embedding. `t_mod` and `out_mod` consist of 2 fully-connected layers (`Linear`) activated by the Sigmoid Linear Unit (`SiLU`) [74]:

$$\texttt{t\_mod} = \texttt{out\_mod} = \texttt{Linear} \to \texttt{SiLU} \to \texttt{Linear}$$

and `y_mod` consists of 3 residual blocks, *i.e.,* $\texttt{y\_mod}(y) = y + \texttt{res\_mod(norm(}y\texttt{))}$, where

$$\texttt{res\_mod} = \texttt{Linear} \to \texttt{SiLU} \to \texttt{Linear} \to \texttt{SiLU} \to \texttt{Linear} \to \texttt{SiLU} \to \texttt{Linear}$$

All `Linear`'s have 128 hidden dimension. We use group normalization [75] for all `norm`. For watermarked generation, we use EDM parameterization[3] [38].

Evaluation We compute the Wasserstein and Sliced Wasserstein distances using the `geomloss`[6] and `ot`[7] packages, respectively. The Maximum Mean Discrepancy (MMD) is based on the popular package `https://github.com/ZongxianLee/MMD_Loss.Pytorch`, which is unlicensed. For watermarked generation, we follow the same evaluation pipeline from EDM[3] [38] by first generating 50,000 watermarked samples and computing the FID w.r.t. the training statistics.

C.1 Additional Results

Tractable variational bound in Equation (8) Figure 9 demonstrates how MDM faithfully captures the variational bound to the negative log-likelihood (NLL) of 2-dimensional GMM.

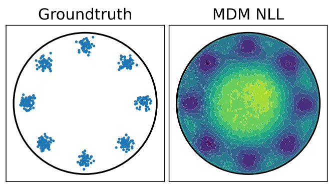

Figure 9: Tractable variational bound by our MDM.

More constrained sets, distributional metrics, & baseline Tables 9 and 10 expand the analysis in Tables 3 and 4 with additional distributional metrics such as Wasserstein-1 ($\mathcal{W}_1$) and Maximum Mean Discrepancy (MMD). Additionally, Table 11 reports the results of hypercube $[0, 1]^d$ constrained set, a special instance of polytopes, and includes additional baseline from Lou and Ermon [25], which approximate the intractable scores in Reflected Diffusion using eigenfunctions toilered specifically to hypercubes, rather than implicit score matching as in Fishman et al. [18]. Consistently, our findings conclude that the MDM is the *only* constrained-based diffusion model that achieves comparable or better performance to DDPM. These results affirm the effectiveness of MDM in generating high-quality samples within constrained settings, making it a reliable choice for constrained generative modeling.

More watermarked samples Figures 10 and 11 provide additional qualitative results on the watermarked samples generated by MDMs.

[4]`https://github.com/openai/guided-diffusion`, released under MIT License.

[5]`https://github.com/louaaron/Reflected-Diffusion`, latest commit (65d05c6) at submission, unlicensed.

[6]`https://github.com/jeanfeydy/geomloss`, released under MIT License.

[7]`https://pythonot.github.io/gen_modules/ot.sliced.html#ot.sliced.sliced_wasserstein_distance`, released under MIT License.

Table 9: Expanded results of **ℓ_2-ball constrained sets**, where we include additional distributional metrics such as $\mathcal{W}_1$ and Maximum Mean Discrepancy (MMD), all computed with 1000 samples and averaged over three trials. Consistently, our findings conclude that the MDM is the *only* constrained-based diffusion model that achieves comparable or better performance to DDPM.

	$d = 2$	$d = 2$	$d = 6$	$d = 8$	d=20
$\mathcal{W}_1$ ↓ (unit: 10^{-2})					
DDPM [2]	0.66 ± 0.15	0.14 ± 0.03	**0.52** ± 0.09	**0.58** ± 0.10	3.45 ± 0.50
Reflected [18]	0.55 ± 0.29	0.46 ± 0.17	3.11 ± 0.40	10.13 ± 0.21	19.42 ± 0.13
MDM (ours)	**0.46** ± 0.07	**0.12** ± 0.04	0.72 ± 0.39	1.05 ± 0.26	**2.63** ± 0.31
MMD ↓ (unit: 10^{-2})					
DDPM [2]	0.67 ± 0.23	**0.23** ± 0.07	**0.37** ± 0.19	0.75 ± 0.24	0.98 ± 0.42
Reflected [18]	0.58 ± 0.46	5.03 ± 1.17	2.34 ± 0.14	28.82 ± 0.66	14.83 ± 0.62
MDM (ours)	**0.52** ± 0.36	0.27 ± 0.19	0.54 ± 0.12	**0.35** ± 0.23	**0.50** ± 0.17
Constraint violation (%) ↓					
DDPM [2]	0.00 ± 0.00	0.00 ± 0.00	8.67 ± 0.87	13.60 ± 0.62	19.33 ± 1.29

Table 10: Expanded results of **simplices constrained sets**.

	$d = 3$	$d = 3$	$d = 7$	$d = 9$	d=20
$\mathcal{W}_1$ ↓ (unit: 10^{-2})					
DDPM [2]	**0.01** ± 0.00	0.02 ± 0.01	**0.03** ± 0.00	**0.05** ± 0.00	**0.11** ± 0.00
Reflected [18]	0.06 ± 0.01	0.12 ± 0.00	0.62 ± 0.08	3.57 ± 0.05	0.98 ± 0.02
MDM (ours)	**0.01** ± 0.00	**0.01** ± 0.01	**0.03** ± 0.00	**0.05** ± 0.00	0.13 ± 0.00
MMD ↓ (unit: 10^{-2})					
DDPM [2]	0.72 ± 0.07	0.72 ± 0.30	0.74 ± 0.10	0.97 ± 0.22	1.12 ± 0.07
Reflected [18]	3.91 ± 0.95	15.12 ± 1.36	16.48 ± 1.04	131.44 ± 2.65	57.90 ± 2.07
MDM (ours)	**0.44** ± 0.16	**0.50** ± 0.26	**0.42** ± 0.08	**0.55** ± 0.13	**0.61** ± 0.03
Constraint violation (%) ↓					
DDPM [2]	0.73 ± 0.12	14.40 ± 1.39	11.63 ± 0.90	27.53 ± 0.57	68.83 ± 1.66

Table 11: Results of **hypercube** $[0, 1]^d$ **constrained sets**.

	$d = 2$	$d = 3$	$d = 6$	$d = 8$	d=20
Sliced Wasserstein ↓ (unit: 10^{-2})					
DDPM [2]	**2.24** ± 1.22	2.17 ± 0.65	2.05 ± 0.41	2.01 ± 0.16	**1.54** ± 0.01
Reflected [25]	3.75 ± 1.20	6.58 ± 1.18	2.77 ± 0.06	3.50 ± 0.69	3.37 ± 0.46
Reflected [18]	19.05 ± 1.51	17.16 ± 0.88	11.90 ± 0.43	7.49 ± 0.13	4.32 ± 0.23
MDM (ours)	3.00 ± 0.72	**1.92** ± 0.81	**1.75** ± 0.17	**1.85** ± 0.34	3.35 ± 0.64
$\mathcal{W}_1$ ↓ (unit: 10^{-2})					
DDPM [2]	**0.07** ± 0.05	0.22 ± 0.07	1.65 ± 0.14	**3.30** ± 0.16	**16.74** ± 0.12
Reflected [25]	0.20 ± 0.12	1.21 ± 0.39	2.53 ± 0.04	4.82 ± 0.42	25.47 ± 0.20
Reflected [18]	4.40 ± 0.57	6.01 ± 0.97	9.34 ± 0.56	9.84 ± 0.24	25.27 ± 0.36
MDM (ours)	0.08 ± 0.03	**0.20** ± 0.07	**1.57** ± 0.08	3.34 ± 0.23	20.59 ± 1.19
MMD ↓ (unit: 10^{-2})					
DDPM [2]	0.27 ± 0.26	0.32 ± 0.14	0.69 ± 0.21	0.81 ± 0.23	0.73 ± 0.07
Reflected [25]	0.92 ± 0.53	3.56 ± 1.31	1.16 ± 0.04	2.09 ± 0.70	2.83 ± 0.58
Reflected [18]	32.26 ± 3.19	26.64 ± 5.07	29.83 ± 1.42	15.84 ± 0.89	7.21 ± 0.68
MDM (ours)	**0.27** ± 0.09	**0.29** ± 0.17	**0.39** ± 0.14	**0.61** ± 0.23	**0.62** ± 0.05
Constraint violation (%) ↓					
DDPM [2]	9.37 ± 0.12	17.57 ± 1.27	41.70 ± 1.30	59.30 ± 1.39	94.47 ± 0.64

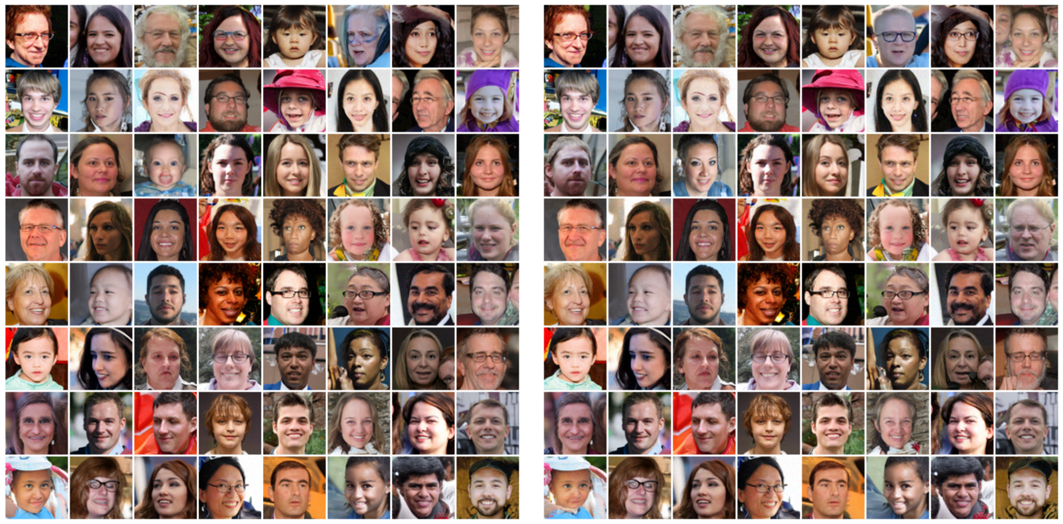

Figure 10: FFHQ 64×64 unconditional watermarked samples generated by (**left**) MDM-proj and (**right**) MDM-dual from the same set of random seeds. Despite the fact that some images, such as the one in the first row and sixth column, were altered possibly due to the change of dual-space distribution (see Figure 7), they look realistic and remain close to the data distribution.

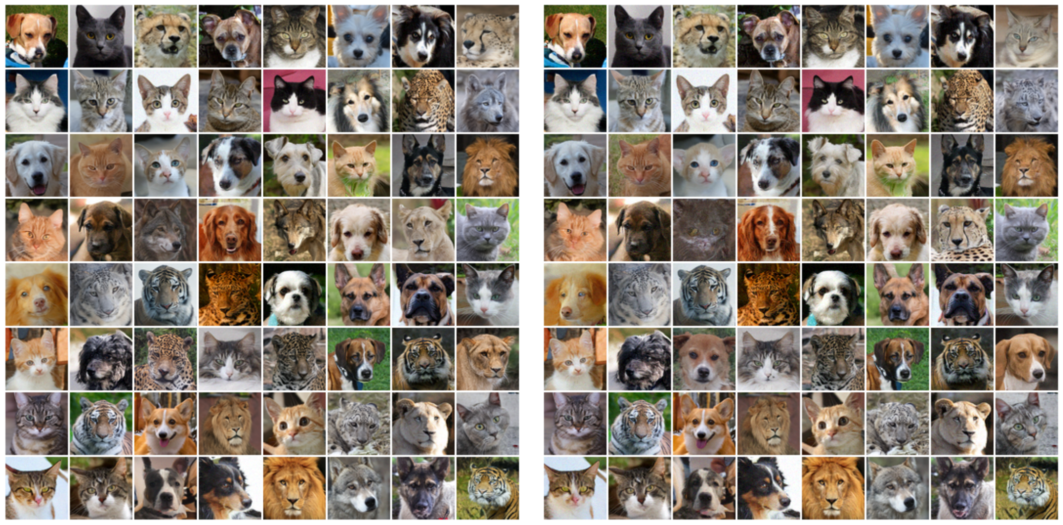

Figure 11: AFHQv2 64×64 unconditional watermarked samples generated by (**left**) MDM-proj and (**right**) MDM-dual from the same set of random seeds. Despite the fact that some images, such as the one in the fifth row and first column, were altered possibly due to the change of dual-space distribution (see Figure 7), they all look realistic and remain close to the data distribution.