Data-Free Training of Diverse Neural Samplers for Constrained Sets

Tilman Burghoff[1], Cornelius V. Braun[1] and Marc Toussaint[1,2]

I. INTRODUCTION

Sampling from constrained sets is a core paradigm for many robotics problems, for example grasping or path planning. Although *optimization* under constraints is a well studied problem, *sampling* under constraints (coined NLP Sampling in [1]) has received less attention.

For unconstrained sampling, diffusion- and flow-based approaches have proven effective in many domains. Recent work by Tong et al. [2] shows the usefulness of diffusion models trained to sample from a constrained set for bi-manual manipulation. However, these models usually need a lot of data for their training. This proves to be a bottleneck, especially in robotics, where datasets are often not readily available and costly to create [3].

Motivated by this, we propose a novel algorithm to train such models "data-free", that is, without using initial training data.

II. APPROACH

We train a diffusion model[3] f_θ to sample uniformly from the constrained set

$$\begin{aligned}\mathscr{M} := \{x \in \mathbb{R}^d \mid g_i(x) \le 0,\ h_j(x) = 0 \\ \text{for } i \in \{1,\dots,m_{ineq}\},\ j \in \{1,\dots,m_{eq}\}\}\,.\end{aligned}$$

Our algorithm works by iteratively training the model to match a distribution of points and then sampling new points from that model to get the next target distribution. To choose values for the parameters $\alpha_{vel}, \alpha_{con}, \gamma_{start}$ and γ_{end}, we perform hyperparameter tuning as described in section III-B.

A. Training

Training is carried out according to the procedure suggested in [6], which corresponded to the lines 5 to 10 in algorithm 1. The central idea of that training procedure is to let the model predict a point x_{pred} in the target distribution directly instead of training it to predict the velocity or the score, but then to consider the velocity $v_{pred} = (x_{pred} - z)/(1-t)$ to compute the loss-function

$$L_{vel}(v, v_{pred}) = \|v - v_{pred}\|^2 = \frac{1}{(1-t)^2}\|x - x_{pred}\|^2. \qquad (1)$$

Algorithm 1 Data Free Model Training

```
1:  function TRAIN($f_\theta, g, h$)
2:      Initialize $D$ with samples drawn from $N(0,1)$
3:      for $k = 1,\dots,k_{max}$ do
4:          for $x \in D$ do
5:              Sample $t \sim U[0,1]$
6:              Sample $e \sim N(0,1)$
7:              $z \leftarrow tx + (1-t)e$
8:              $v \leftarrow (x-z)/(1-t)$
9:              $x_{pred} \leftarrow f_\theta(z,t)$
10:             $v_{pred} \leftarrow (x_{pred} - z)/(1-t)$
11:             $L \leftarrow \alpha_{vel} L_{vel}(v, v_{pred}) + \alpha_{con} L_{con}(x_{pred}, t)$
12:             Update $\theta$ with gradient descent on $L$
13:         $D \leftarrow$ CREATEDATASET($f_\theta, k$)
14: function CREATEDATASET($f_\theta, k$)
15:     $\gamma \leftarrow \gamma_{start} + \frac{k}{k_{max}}(\gamma_{end} - \gamma_{start})$
16:     $D \leftarrow \emptyset$
17:     while $|D| < (1-\gamma)n$ do
18:         $x \leftarrow$ SAMPLE($f_\theta$)
19:         if $L_{con}(x) < s$ then
20:             $D \leftarrow D \cup \{x\}$
21:     $\hat{D} \leftarrow \{x \mid x \sim N(0,1)\}$ such that $|\hat{D}| = \gamma|D|$
22:     return $D \cup \hat{D}$
```

We add a loss that steers the samples towards $\mathscr{M}$ by summing the squared constraint violations

$$L_{con}(x,t) = \frac{1}{(1-t)^2}\left(\sum_{i=0}^{m_{ineq}} [g_i(x)]_+^2 + \sum_{j=0}^{m_{eq}} h_j(x)^2\right). \qquad (2)$$

This loss is scaled by $\frac{1}{(1-t)^2}$ to ensure that it has the same scale as the velocity loss (1). To prevent division by zero, we follow [6] and clip $1-t$ to be at least 0.05 in the denominator in the constraint loss (2) and during the velocity computation in algorithm 1 line 8 and line 10.

Previous research cautions against using this loss due to the possibility of mode collapse [7]. Our algorithm addresses this challenge with the resampling step, as detailed in section II-B.

B. Sampling

After training the model with the previous samples, we sample n new points for the next iteration. We take a fraction γ of the new points to be univariate gaussian distributed, i.e. noise. This encourages the model to explore and it prevents mode collapse. This is nessecary, as without noise, projecting onto a single point in $\mathscr{M}$ would be an optimal solution. Empirically, the effect of low noise can be seen in Figure 1. In the figure the middle disk slowly vanishes, since the model

This version contains minor editorial revisions for clarity to the extended abstract submitted to the 2nd German Robotics Conference.

[1]Technische Universität Berlin, Learning & Intelligent Systems Lab {t.burghoff, braun, toussaint}@tu-berlin.de

[2]Robotics Institute Germany (RIG)

[3]or a flow model, the difference only comes into play during inference [4], [5]. Unless stated, we use the term diffusion model to refer to both methods.

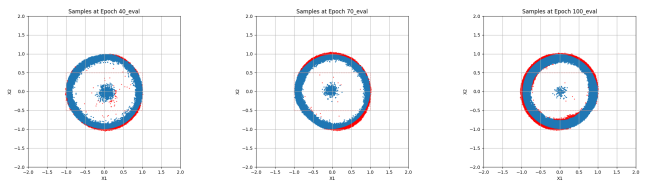

Fig. 1: This figure shows the effect of low noise. The middle circle almost vanishes since the model tends to focus weight on the outer ring.

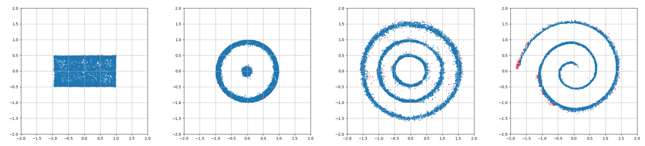

Fig. 2: The resulting distribution sampled using flow matching after training the model for 50 epochs.

is mostly trained on its own data which results in a feedback loop where strong modes get reinforced. We further discuss the effect of noise in section III-B.

The rest of the samples are generated by querying the model, using either the Euler method for flow matching or the Euler-Maruyama method for diffusion. Qualitatively, flow matching leads to higher quality samples, as demonstrated by lower constraint violations. This is especially important on manifolds, where flow matching achieves a lower spread along the normal. Conversely, visual inspection suggest that diffusion explores the space more.

In practice, it is unlikely that the model generates a large number of valid samples in the early stages of training. We therefore recommend limiting the iterations of the loop in algorithm 1 line 17. For our experiments, we generate at maximum $10n$ samples.

III. Experiments and Results

A. Two-dimensional Distributions

We use different 2-dimensional distributions to visually examine the performance of the method. For each distribution in table I we train a dense neural network with 4 hidden layers and 128 neurons each using the parameters (3) for our algorithm. The model is evaluated by using flow matching to sample a distribution. In table I, we use

$$\mathrm{Slack}(x) := \sum_{i=0}^{m_{ineq}} [g_i(x)]_+ + \sum_{j=0}^{m_{eq}} |h_j(x)| \, ,$$

Name	Constraints	Succ	AvSl
Rectangle	$\lvert x_1 \rvert \leq 1,\ \lvert x_2 \rvert \leq 0.5$	0.964	3.2e−4
Shield	$\min \left\{ \begin{array}{l} \lVert x \rVert - 0.25, \\ [0.75 - \lVert x \rVert]_+ + [\lVert x \rVert - 1]_+ \end{array} \right\} \leq 0$	0.993	1e−4
Three Rings	$\min_i \lvert \lVert x \rVert - r_i \rvert \leq 0.1$ for $r_i = 0.5i,\ i \in \{1,2,3\}$	0.903	4.2e−4
Spiral	$\min_i 0.1 + 0.1(\theta + 2\pi i) = \lVert x \rVert$ $i \in \{0,1,2\},\ \theta = \mathrm{atan2}(x_2, x_1)$	0.973	2.5e−4

TABLE I: The four 2d Problems and their performance

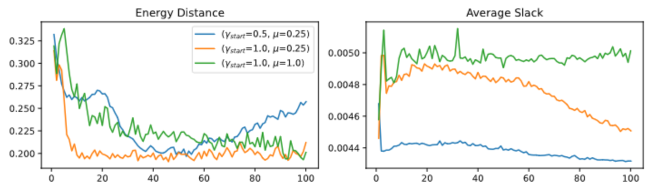

Fig. 3: Energy distance and average slack while training model in section III-B for $d = 50$ for 100 epochs.

to report the average slack $\mathrm{AvSl} := \mathbb{E}[\mathrm{Slack}(x)]$ and the success rate $\mathrm{Succ} := \mathbb{P}(\mathrm{Slack}(x) < s)$ for $s = 0.001$, averaged over 5 runs. Figure 2 shows the sampled distributions.

B. Hyperparameter Tuning

To find the optimal parameters, we want to challenge the model with a problem that contains multiple low-dimensional manifolds. To that end, we sample 5 two-dimensional disks embedded in $\mathbb{R}^2 \times \{0\}^{d-2}$ for $d \in \{10, 50\}$ and rotate each one with a random orthonormal matrix $O_i \in \mathrm{SO}(d)$. This setup also allows us to easily generate ground truth samples, enabling us to evaluate the sample quality of the model using the energy distance [8].

The best performing parameters for this setup are

$$\alpha_v = \alpha_{con} = 1,\ \gamma_{start} = 1.0 \text{ and } \gamma_{end} = 0.25\,. \qquad (3)$$

with an energy distance of 0.072 ± 0.015 and an average slack of 0.002 ± 0.0 for $d = 10$ and $ED = 0.202 \pm 0.041$ and an average slack 0.005 ± 0.0 for $d = 50$.

The noise schedule given by $\gamma_{start}, \gamma_{end}$ has an important effect on the models performance. Figure 3 shows its effect for three example parameters. We can see that for low initial noise, the slack goes down fast, while the energy distance gets worse at the end. This indicates mode collapse. In comparison, when we use too much noise, the model never learns to get close to the manifolds and the average slack stays high.

IV. Discussion

We propose a novel algorithm for data-free training of diffusion models. Although we only evaluated it on small examples, we believe it shows promising performance. As demonstrated, the choice of noise schedule is crucial to prevent mode collapse while allowing the model to accurately learn the desired distribution.

An attractive quality is that the algorithm only slightly modifies the training of the model without changing the architecture or inference procedure. This enables it to work together with a wide range of solutions. However, even using simple networks and inference, we are able to achieve good performance.

Acknowledgment

The authors thank Paula Cordero Encinar for the insightful discussions on sampling and Henning Burghoff for the helpful editorial suggestions. This research was funded by the Amazon Fulfillment Technologies and Robotics team. This work has been supported by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG).

References

[1] M. Toussaint, C. V. Braun, and J. Ortiz-Haro, *NLP Sampling: Combining MCMC and NLP Methods for Diverse Constrained Sampling*, Jul. 2024.

[2] H. Tong, Y. Zhang, S. Lueth, and G. Chalvatzaki, *Adaptive Diffusion Constrained Sampling for Bimanual Robot Manipulation*, Oct. 2025.

[3] K. Goldberg, "Good old-fashioned engineering can close the 100,000-year 'data gap' in robotics," *Science Robotics*, vol. 10, no. 105, Aug. 2025, ISSN: 2470-9476.

[4] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie, "SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., ser. Lecture Notes in Computer Science, vol. 15135, Cham: Springer Nature Switzerland, Sep. 2024, pp. 23–40, ISBN: 978-3-031-72980-5.

[5] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, "Stochastic Interpolants: A Unifying Framework for Flows and Diffusions," *Journal of Machine Learning Research*, vol. 26, no. 209, Sep. 2025, ISSN: 1533-7928.

[6] T. Li and K. He, *Back to Basics: Let Denoising Generative Models Denoise*, Nov. 2025.

[7] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint, "Structured deep generative models for sampling on constraint manifolds in sequential manipulation," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Nov. 2022, pp. 213–223.

[8] G. J. Szekely and M. L. Rizzo, *The Energy of Data and Distance Correlation*. New York: Chapman and Hall/CRC, Feb. 2023, ISBN: 978-0-429-15715-8.