Diffusion Generative Inverse Design

Marin Vlastelica [1 2] **Tatiana López-Guevara** [2] **Kelsey Allen** [2] **Peter Battaglia** [2] **Arnaud Doucet** [2]
Kimberly Stachenfeld [2 3]

Abstract

Inverse design refers to the problem of optimizing the input of an objective function in order to enact a target outcome. For many real-world engineering problems, the objective function takes the form of a simulator that predicts how the system state will evolve over time, and the design challenge is to optimize the initial conditions that lead to a target outcome. Recent developments in learned simulation have shown that graph neural networks (GNNs) can be used for accurate, efficient, differentiable estimation of simulator dynamics, and support high-quality design optimization with gradient- or sampling-based optimization procedures. However, optimizing designs from scratch requires many expensive model queries, and these procedures exhibit basic failures on either non-convex or high-dimensional problems. In this work, we show how denoising diffusion models (DDMs) can be used to solve inverse design problems efficiently and propose a particle sampling algorithm for further improving their efficiency. We perform experiments on a number of fluid dynamics design challenges, and find that our approach substantially reduces the number of calls to the simulator compared to standard techniques.

1. Introduction

Substantial improvements to our way of life hinge on devising solutions to engineering challenges, an area in which Machine Learning (ML) advances is poised to provide positive real-world impact. Many such problems can be formulated as designing an object that gives rise to some desirable physical dynamics (e.g. designing an aerodynamic car or a watertight vessel). Here we are using ML to accelerate this design process by learning both a forward model of the dynamics and a distribution over the design space.

Prior approaches to ML-accelerated design have used neural networks as a differentiable forward model for optimization (Challapalli et al., 2021; Christensen et al., 2020; Gómez-Bombarelli et al., 2018). We build on work in which the forward model takes the specific form of a GNN trained to simulate fluid dynamics (Allen et al., 2022). Since the learned model is differentiable, design optimization can be accomplished with gradient-based approaches (although these struggle with zero or noisy gradients and local minima) or sampling-based approaches (although these fare poorly in high-dimensional design spaces). Both often require multiple expensive calls to the forward model. However, generative models can be used to propose plausible designs, thereby reducing the number of required calls (Forte et al., 2022; Zheng et al., 2020; Kumar et al., 2020).

In this work, we use DDMs to optimize designs by sampling from a target distribution informed by a learned data-driven prior. DDMs have achieved extraordinary results in image generation (Song et al., 2020a;b; Karras et al., 2022; Ho et al., 2020), and has since been used to learn efficient planners in sequential decision making and reinforcement learning (Janner et al., 2022; Ajay et al., 2022), sampling on manifolds (De Bortoli et al., 2022) or constrained optimization formulations (Graikos et al., 2022). Our primary contribution is to consider DDMs in the setting of physical problem solving. We find that such models combined with continuous sampling procedures enable to solve design problems orders of magnitude faster than off-the-shelf optimizers such as CEM and Adam. This can be further improved by utilizing a particle sampling scheme to update the base distribution of the diffusion model which by cheap evaluations (few ODE steps) with a learned model leads to better designs in comparison to vanilla sampling procedures. We validate our findings on multiple experiments in a particle fluid design environment.

2. Method

Given some task specification $\boldsymbol{c}$, we have a target distribution of designs $\pi(\boldsymbol{x})$ which we want to optimize w.r.t. $\boldsymbol{x}$. To

[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany [2]Google DeepMind, London, UK [3]Columbia University, New York, NY. Correspondence to: Marin Vlastelica <marin.vlastelica@tue.mpg.de>, Kimberly Stachenfeld <stachenfeld@deepmind.com>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling

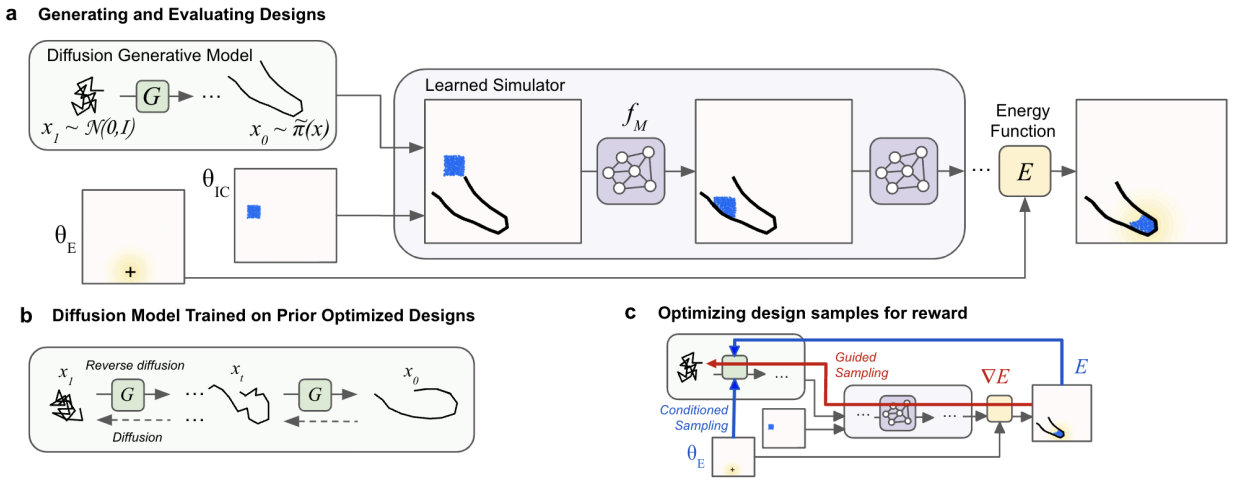

Figure 1. (a) Given initial conditions governed by θ_{IC}, energy function parameters θ_E, and learned GNN dynamics model f_M, design samples $\boldsymbol{x}$ from the diffusion model are assigned a cost $E(\boldsymbol{x})$. (b) Schematic of the DDM training (c) Gradients ∇E and conditioning set (θ_E and E) inform energy and conditional guidance, resp.

simplify notation, we do not emphasize the dependence of π on $\boldsymbol{c}$. This distribution is a difficult object to handle, since a highly non-convex cost landscape might hinder efficient optimization. We can capture prior knowledge over 'sensible' designs in form of a prior distribution $p(\boldsymbol{x})$ learned from existing data. Given a prior, we may sample from the distribution

$$\tilde{\pi}(\boldsymbol{x}) \propto p(\boldsymbol{x})\pi(\boldsymbol{x}), \tag{1}$$

which in this work is achieved by using a diffusion method with guided sampling. The designs will subsequently be evaluated by a learned forward model comprised of a pre-trained GNN simulator and a reward function (Allen et al., 2022; Pfaff et al., 2021; Sanchez-Gonzalez et al., 2020) (see Appendix A).

Let $E : \mathbb{X} \mapsto \mathbb{R}$ be the cost (or "energy") of a design $\boldsymbol{x} \in \mathbb{X}$ for a specific task $\boldsymbol{c}$ under the learned simulator (dependence of E on $\boldsymbol{c}$ is omitted for simplicity). The target distribution of designs $\pi(\boldsymbol{x})$ is defined by the Boltzmann distribution

$$\pi(\boldsymbol{x}) := \frac{1}{Z} \exp\left(-\frac{E(\boldsymbol{x})}{\tau}\right), \tag{2}$$

where Z denotes the unknown normalizing constant and τ a temperature parameter. As $\tau \to 0$, this distribution concentrates on its modes, that is on the set of the optimal designs for the cost $E^{\boldsymbol{c}}(\boldsymbol{x})$. Direct methods to sample from $\pi(\boldsymbol{x})$ rely on expensive Markov chain Monte Carlo techniques or variational methods minimizing a reverse KL criterion.

We will rely on a data-driven prior learned by the diffusion model from previous optimization attempts. We collect optimization trajectories of designs for different task parametrizations $\boldsymbol{c}$ using Adam (Kingma & Ba, 2015) or CEM (Rubinstein, 1999) to optimize $\boldsymbol{x}$. Multiple entire optimization trajectories of designs are included in the training set for the generative model, providing a mix of design quality. These optimization trajectories are initialized to flat tool(s) below the fluid (see Figure 5), which can be more easily shaped into successful tools than a randomly initialized one. Later, when we compare the performance of the DDM to Adam and CEM, we will be using randomly initialized tools for Adam and CEM, which is substantially more challenging.

2.1. Diffusion generative models

We use DDMs to fit $p(\boldsymbol{x})$ (Ho et al., 2020; Song et al., 2020b). The core idea is to initialize using training data $\boldsymbol{x}_0 \sim p$, captured by a diffusion process $(\boldsymbol{x}_t)_{t\in[0,1]}$ defined by

$$\mathrm{d}\boldsymbol{x}_t = -\beta_t \boldsymbol{x}_t \mathrm{d}t + \sqrt{2\beta_t}\mathrm{d}\boldsymbol{w}_t, \tag{3}$$

where $(\boldsymbol{w}_t)_{t\in[0,1]}$ denotes the Wiener process. We denote by $p_t(\boldsymbol{x})$ the distribution of $\boldsymbol{x}_t$ under (3). For β_t large enough, $p_1(\boldsymbol{x}) \approx \mathcal{N}(\boldsymbol{x}; 0, I)$. The time-reversal of (3) satisfies

$$\mathrm{d}\boldsymbol{x}_t = -\beta_t[\boldsymbol{x}_t + 2\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)]\mathrm{d}t + \sqrt{2\beta_t}\mathrm{d}\boldsymbol{w}_t^-, \tag{4}$$

where $(\boldsymbol{w}_t^-)_{t\in[0,1]}$ is a Wiener process when time flows backwards from $t = 1$ to $t = 0$, and $\mathrm{d}t$ is an infinitesimal negative timestep. By initializing (4) using $\boldsymbol{x}_1 \sim p_1$, we obtain $\boldsymbol{x}_0 \sim p$. In practice, the generative model is obtained by sampling an approximation of (4), replacing $p_1(\boldsymbol{x})$ by $\mathcal{N}(\boldsymbol{x}; 0, I)$ and the intractable score $\nabla_x \log p_t(\boldsymbol{x})$ by $s_\theta(\boldsymbol{x}, t)$. The score estimate $s_\theta(\boldsymbol{x}, t)$ is learned by denoising score matching, i.e. we use the fact that $\nabla_x \log p_t(\boldsymbol{x}) = \int \nabla_x \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_t)\mathrm{d}\boldsymbol{x}_0$ where $p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_0, \sqrt{1-\alpha_t}I)$ is the transition density of (3), α_t being a function of $(\beta_s)_{s\in[0,t]}$ (Song et al., 2020b). It follows straightforwardly that the

score satisfies $\nabla_x \log p_t(\boldsymbol{x}) = -\mathbb{E}[\epsilon|\boldsymbol{x}_t = \boldsymbol{x}]/\sqrt{1-\alpha_t}$ for $\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\epsilon$. We then learn the score by minimizing

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x}_0 \sim p, t \sim \mathcal{U}(0,1), \epsilon \sim \mathcal{N}(0,I)} \|\epsilon_\theta(\boldsymbol{x}_t, t) - \epsilon\|^2, \quad (5)$$

where $\epsilon_\theta(\boldsymbol{x}, t)$ is a denoiser estimating $\mathbb{E}[\epsilon|\boldsymbol{x}_t = \boldsymbol{x}]$. The score function $s_\theta(\boldsymbol{x}, t) \approx \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ is obtained using

$$s_\theta(\boldsymbol{x}, t) = -\frac{\epsilon_\theta(\boldsymbol{x}, t)}{\sqrt{1-\alpha_t}}. \quad (6)$$

Going forward, ∇ refers to $\nabla_{\boldsymbol{x}}$ unless otherwise stated. We can also sample from $p(\boldsymbol{x})$ using an ordinary differential equation (ODE) developed in (Song et al., 2020b).

Let us define $\overline{\boldsymbol{x}}_t = \boldsymbol{x}_t/\sqrt{\alpha_t}$ and $\sigma_t = \sqrt{1-\alpha_t}/\sqrt{\alpha_t}$. Then by initializing $\boldsymbol{x}_1 \sim \mathcal{N}(0, I)$, equivalently $\overline{\boldsymbol{x}}_1 \sim \mathcal{N}(0, \alpha_1^{-1} I)$ and solving backward in time

$$\mathrm{d}\overline{\boldsymbol{x}}_t = \epsilon_\theta^{(t)}\left(\frac{\overline{\boldsymbol{x}}_t}{\sqrt{\sigma_t^2 + 1}}\right) \mathrm{d}\sigma_t, \quad (7)$$

then $\boldsymbol{x}_0 = \sqrt{\alpha_t}\,\overline{\boldsymbol{x}}_0$ is an approximate sample from $p(\boldsymbol{x})$.

2.2. Approximately sampling from target $\tilde{\pi}(\boldsymbol{x})$

We want to sample $\tilde{\pi}(\boldsymbol{x})$ defined in (1) where $p(\boldsymbol{x})$ can be sampled from using the diffusion model. We describe two possible sampling procedures with different advantages for downstream optimization.

Energy guidance. Observe that

$$\tilde{\pi}_t(\boldsymbol{x}_t) = \int \tilde{\pi}(\boldsymbol{x}_0) p(\boldsymbol{x}_t|\boldsymbol{x}_0) \mathrm{d}\boldsymbol{x}_0,$$

and the gradient satisfies

$$\nabla \log \tilde{\pi}_t(\boldsymbol{x}_t) = \nabla \log p_t(\boldsymbol{x}_t) + \nabla \log \pi_t(\boldsymbol{x}_t),$$

where $\pi_t(\boldsymbol{x}_t) = \int \pi(\boldsymbol{x}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_t) \mathrm{d}\boldsymbol{x}_0$. We approximate this term by making the approximation

$$\begin{aligned} \hat{\boldsymbol{x}}(\boldsymbol{x}_t, t) &= \underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}}\right)}_{\text{“estimated } \boldsymbol{x}_0\text{”}}, \\ \pi_t(\boldsymbol{x}_t) &\approx \pi(\hat{\boldsymbol{x}}_t(\boldsymbol{x}_t, t)). \end{aligned} \quad (8)$$

Now, by (6), and the identity $\nabla \log \pi(\boldsymbol{x}) = -\tau^{-1}\nabla E(\boldsymbol{x})$, we may change the reverse sampling procedure by a modified denoising vector

$$\tilde{\epsilon}_\theta(\boldsymbol{x}_t, t) = \epsilon_\theta(\boldsymbol{x}_t, t) + \lambda\tau^{-1}\sqrt{1-\alpha_t}\nabla E(\hat{\boldsymbol{x}}(\boldsymbol{x}_t, t)), \quad (9)$$

with λ being an hyperparameter. We defer the results on energy guidance Appendix E.

Algorithm 1 Particle optimization of base distribution.

input energy function E, diffusion generative model p_θ, temperature τ, noise scale σ, rounds K.
$\mathbb{S}_1^0 = \{\boldsymbol{x}_1^i\}_{i=1}^N$ for $\boldsymbol{x}_1^i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I)$
$\mathbb{S}_0 = \emptyset, \mathbb{S}_1 = \emptyset$ # $t = 0$ and $t = 1$ sample sets
for $k \in \{0 \dots K\}$ **do**
  Compute $\mathbb{S}_0^k = \{\boldsymbol{x}_0^i\}_{i=1}^N$ from $\mathbb{S}_1^k$ by solving reverse ODE in Eq. (7).
  $\mathbb{S}_0 = \mathbb{S}_0 \cup \mathbb{S}_0^k, \mathbb{S}_1 = \mathbb{S}_1 \cup \mathbb{S}_1^k$
  Compute normalized importance weights $\mathbb{W} = \left\{w \mid w \propto \exp\left(-\frac{E(\boldsymbol{x}_0)}{\tau}\right), \boldsymbol{x}_0 \in \mathbb{S}_0\right\}$
  Set $\overline{\mathbb{S}}_1^{k+1} = \{\overline{\boldsymbol{x}}_1^i\}_{i=1}^{|\mathbb{S}_1|}$ for $\overline{\boldsymbol{x}}_1^i \overset{\text{i.i.d.}}{\sim} \sum_{i=1}^{|\mathbb{S}_1|} w^i \delta_{\boldsymbol{x}_1^i}(\boldsymbol{x})$
  Set $\mathbb{S}_1^{k+1} = \{\tilde{\boldsymbol{x}}_1^i\}_{i=1}^{|\mathbb{S}_1|}$ for $\tilde{\boldsymbol{x}}_1^i \sim \mathcal{N}(\boldsymbol{x}; \overline{\boldsymbol{x}}_1^i, \sigma^2 I)$
end for
return $\arg\min_{\boldsymbol{x} \in \mathbb{S}_0} E(\boldsymbol{x})$

Conditional guidance. Similarly to *classifier-free* guidance (Ho & Salimans, 2022), we explore conditioning on cost (energy) $\boldsymbol{e}$ and task $\boldsymbol{c}$. A modified denoising vector in the reverse process follows as a combination between the denoising vector of a conditional denoiser ϵ_ϕ and unconditional denoiser ϵ_θ

$$\tilde{\epsilon}(\boldsymbol{x}_t, \boldsymbol{c}, \boldsymbol{e}, t) = (1+\lambda)\epsilon_\phi(\boldsymbol{x}_t, \boldsymbol{c}, \boldsymbol{e}, t) - \lambda\epsilon_\theta(\boldsymbol{x}_t, t), \quad (10)$$

where ϵ_ϕ is learned by conditioning on $\boldsymbol{c}$ and cost $\boldsymbol{e}$ from optimization trajectories. In our experiments we shall choose $\boldsymbol{c}$ to contain the design cost percentile and target goal destination θ_E for fluid particles (Figure 1c).

2.3. A modified base distribution through particle sampling

Our generating process initializes samples at time $t = 1$ from $\mathcal{N}(\boldsymbol{x}; 0, I) \approx p_1(\boldsymbol{x})$. The reverse process with modifications from subsection 2.2 provides approximate samples from $\tilde{\pi}(\boldsymbol{x})$ at $t = 0$. However, as we are approximately solving the ODE of an approximate denoising model with an approximate cost function, this affects the quality of samples with respect to E[1]. Moreover, “bad” samples from $\mathcal{N}(\boldsymbol{x}; 0, I)$ are hard to correct by guided sampling.

To mitigate this, instead of using samples from $\mathcal{N}(\boldsymbol{x}; 0, I)$ to start the reverse process of $\tilde{\pi}(\boldsymbol{x})$, we use a multi-step particle sampling scheme which evaluates the samples $\{\boldsymbol{x}_1^i\}_{i=1}^N$ by a rough estimate of the corresponding $\{\boldsymbol{x}_0^i\}_{i=1}^N$ derived from a few-step reverse process and evaluation with E. The particle procedure relies on re-sampling from a weighted particle approximation of $\pi(\boldsymbol{x})$ and then perturbing the resampled particles, see 1. This heuristic does not provide samples from $\tilde{\pi}$ but we found that it provides samples of lower energy samples across tasks. However, with N samples in each of the k rounds, it still requires $\mathcal{O}(Nk)$ evaluations of E, which may be prohibitively expensive depending on the choice of E.

[1]Ideally at test time we would evaluate the samples with the ground-truth dynamics model, but we have used the approximate GNN model due to time constraints on the project.

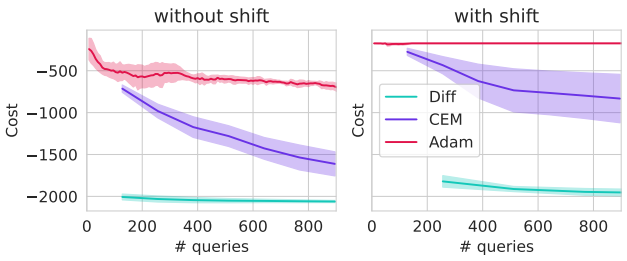

Figure 2. Performance of the different optimization methods in the angle optimization task. We observe that the diffusion generative model requires a small number of model queries, whereas Adam in comparison requires many expensive model queries.

3. Experiments

We evaluate our approach on a 2D fluid particle environment with multiple tasks of varying complexity, which all involve shaping the tool (Figure 1a, black lines) such that the fluid particles end up in a region of the scene that minimizes the cost E (Allen et al., 2022). As baselines, we use the Adam optimizer combined with a learned model of the particles and the CEM optimizer which also optimizes with a learned model. For all experiments we have used a simple MLP as the denoising model and GNN particle simulation model for evaluation of the samples.

The first task is a simple "Contain" task with a single source of water particles, a goal position above the floor specified by $\boldsymbol{c} = (x, y)$, and an articulated tool with 16 joints whose angles are optimized to contain the fluid near the goal (see Allen et al. (2022)). In Figure 2a, we see that both the Adam optimizer and CEM are able to optimize the task. However with training a prior distribution on optimization trajectories and guided sampling we are able to see the benefits of having distilled previous examples of optimized designs into our prior, and achieve superior performance with fewer samples in unseen tasks sampled in-distribution.

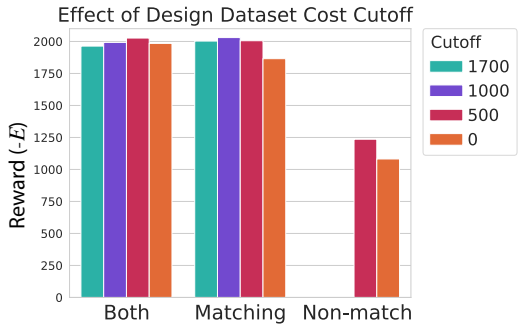

Figure 3. Generative models were trained on a dataset of designs produced from CEM- and Adam-optimized designs on either of two tasks Matching, Non-matching, or Both. Designs in the train dataset were filtered to have costs below the specified cutoff.

If we modify the task by introducing a parameter controlling x, y shift parameter, we observe that Adam fails. This is because there are many values of the shift for which the tool makes no contact with the fluid (see Figure 2b), and therefore gets no gradient information from E. We provide results for more complex tasks in Appendix C (Figure 5). Overall, we find that this approach is capable of tackling a number of different types of design challenges, finding effective solutions when obstacles are present, for multi-modal reward functions, and when multiple tools must be coordinated.

3.1. Dataset quality impact

We analyzed how the model performs with conditional guidance when trained on optimization trajectories of CEM that optimize the same task (matching), a different task (non-matching), or a mix of tasks (Figure 3). The two tasks were "Contain" (described above) and "Ramp" (transport fluid to the far lower corner). Unsurprisingly, the gap between designs in the training dataset and the solution for the current task has a substantial impact on performance. We also control the quality of design samples by filtering out samples above a certain cost level for the $\boldsymbol{c}$ with which they were generated. Discarding bad samples from the dataset does not improve performance beyond a certain point: best performance is obtained with some bad-performing designs in the dataset. Intuitively, we believe this is because limiting the dataset to a small set of optimized designs gives poor coverage of the design space, and therefore generalizes poorly even to in-distribution test problems. Further, training the generative model on samples from optimization trajectories of only a non-matching task has a substantial negative impact on performance. We expect the energy guidance not to suffer from the same transfer issues as conditional guidance, since more information about the task is given through the energy function. Since we indeed obtain data to fit $p(\boldsymbol{x})$ from Adam and CEM runs, why is diffusion more efficient? We discuss this in Appendix B.

3.2. Particle optimization in base distribution

We also observe performance improvements by using the particle search scheme from subsection 2.3, see Figure 4.

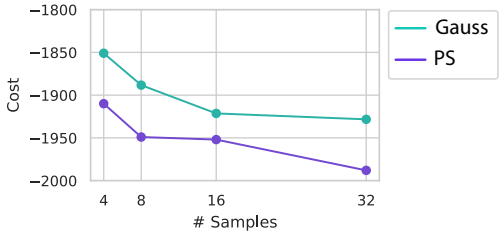

Figure 4. Gaussian base distribution vs. particle sampling (PS).

We hypothesize that the reason for this is because of function approximation. Since we are dealing with approximate scores, it is hard for the learned generative model to pinpoint the optima, therefore a sampling-based search approach helps. We note that the evaluation of a sample with E requires solving the ODE for sample $\boldsymbol{x}_1$, we use a 1-step reverse process making use of the relation in (8). Consequently, we can expect that linearizing the sampling will improve the particle search.

4. Conclusion

In this work we have demonstrated the benefits of using diffusion generative models in simple inverse designs tasks where we want to sample from high probability regions of a target distribution $\pi(\boldsymbol{x})$ defined via E, while having access to optimization data. We analyzed energy-based and conditional guidance where the energy function involves rolling out a GNN. We find that energy guidance is a viable option, but conditional guidance works better in practice, and that performance depends heavily on the generative model's training data. Finally, we have introduced particle search in the base distribution as a means to improve quality of the samples and demonstrated this on multiple tasks.

5. Acknowledgments

We would like to thank Conor Durkan, Charlie Nash, George Papamakarios, Yulia Rubanova, and Alvaro Sanchez-Gonzalez for helpful conversations about the project. We would also like to thank Alvaro Sanchez-Gonzalez for comments on the manuscript.

References

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision-making? In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Allen, K. R., Lopez-Guevara, T., Stachenfeld, K. L., Sanchez-Gonzalez, A., Battaglia, P. W., Hamrick, J. B., and Pfaff, T. Physical design using differentiable learned simulators. *CoRR*, abs/2202.00728, 2022. URL https://arxiv.org/abs/2202.00728.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Challapalli, A., Patel, D., and Li, G. Inverse machine learning framework for optimizing lightweight metamaterials. *Materials & Design*, 208:109937, 2021.

Christensen, T., Loh, C., Picek, S., Jakobović, D., Jing, L., Fisher, S., Ceperic, V., Joannopoulos, J. D., and Soljačić, M. Predictive and generative machine learning models for photonic crystals. *Nanophotonics*, 9(13):4183–4192, 2020.

De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. Riemannian score-based generative modelling. In *Advances in Neural Information Processing Systems*, 2022.

Forte, A. E., Hanakata, P. Z., Jin, L., Zari, E., Zareei, A., Fernandes, M. C., Sumner, L., Alvarez, J. T., and Bertoldi, K. Inverse design of inflatable soft membranes through machine learning. *Advanced Functional Materials*, 32, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 02 2018.

Graikos, A., Malkin, N., Jojic, N., and Samaras, D. Diffusion models as plug-and-play priors. In *Advances in Neural Information Processing Systems*, 2022.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kumar, S., Tan, S., Zheng, L., and Kochmann, D. M. Inverse-designed spinodoid metamaterials. *npj Computational Materials*, 6:1–10, 2020.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Rubinstein, R. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1:127–190, 1999.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020b.

Zheng, L., Kumar, S., and Kochmann, D. M. Data-driven topology optimization of spinodoid metamaterials with seamlessly tunable anisotropy. *ArXiv*, abs/2012.15744, 2020.

Appendix for Diffusion Generative Inverse Design

A. Learned simulation with graph neural networks

As in Allen et al. (2022), we rely on the recently developed MESHGNN model (Pfaff et al., 2021), which is an extension of the GNS model for particle simulation (Sanchez-Gonzalez et al., 2020). MESHGNN is a type of message-passing graph neural network (GNN) that performs both edge and node updates (Battaglia et al., 2018; Gilmer et al., 2017), and which was designed specifically for physics simulation.

We consider simulations over physical states represented as graphs $G \in \mathcal{G}$. The state $G = (V, E)$ has nodes V connected by edges E, where each node $v \in V$ is associated with a position $\mathbf{u}_v$ and additional dynamical quantities $\mathbf{q}_v$. In this environment, each node corresponds to a particle and edges are computed dynamically based on proximity.

The simulation dynamics on which the model is trained are given by a "ground-truth" simulator which maps the state $\mathcal{G}^t$ at time t to the state $\mathcal{G}^{t+1}$ at time $t + \Delta t$. The simulator can be applied iteratively over K time steps to yield a trajectory of states, or a "rollout," which we denote $(G^{t_0}, ..., G^{t_K})$. The GNN model M is trained on these trajectories to imitate the ground-truth simulator f_S. The learned simulator f_M can be similarly applied to produce rollouts $(\tilde{G}^{t_0}, \tilde{G}^{t_1}, ..., \tilde{G}^{t_K})$, where $\tilde{G}^{t_0} = G^{t_0}$ represents initial conditions given as input.

B. Further discussion on results

We trained the diffusion model on data generated from Adam and CEM optimization runs and noticed an improvement over Adam and CEM on the evaluation tasks. The reason for this is that both Adam and CEM need good initializations to solve the tasks efficiently, for example in Figure 2 for each run the initial design for Adam has uniformly sampled angles and x, y coordinates within the bounds of the environment, which would explain why we obtain worse results *on average* for Adam than Allen et al. (2022). Similarly, for CEM we use a Gaussian sampling distribution which is initialized with zero mean and identity covariance. If most of the density of the initial CEM distribution is not concentrated near the optimal design, then CEM will require many samples to find it.

In comparison, the diffusion model learns good initializations of the designs through $p(\boldsymbol{x})$ which can further be improved via guided sampling, as desired.

C. Particle environments

For evaluation, we consider similar fluid particle-simulation environments as in Allen et al. (2022). The goal being to design a 'tool' that brings the particles in a specific configuration. We defined the energy as the radial basis function

$$E(\boldsymbol{x}) = \sum_{p \in \mathbb{P}} \exp\left(-\frac{\|\boldsymbol{x}_p^t - \boldsymbol{x}_p^\star\|}{\sigma}\right),$$

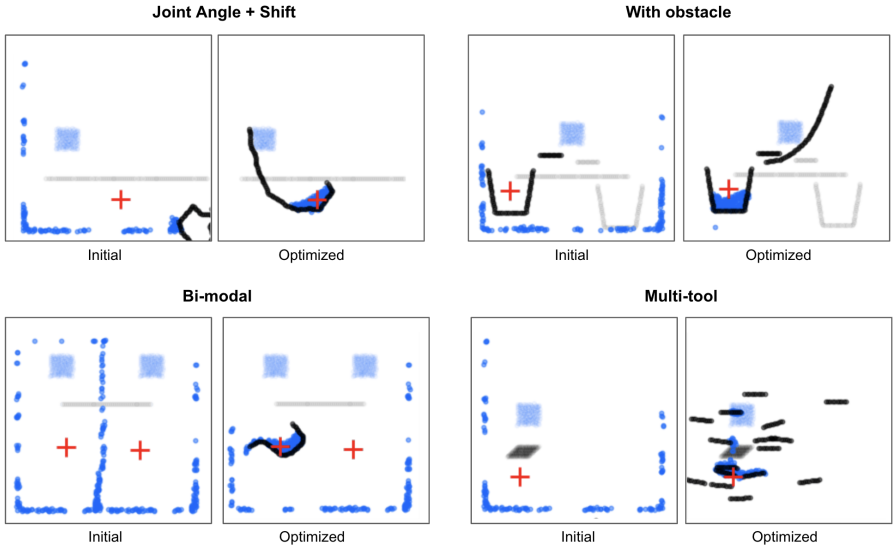

Figure 5. Examples of guided-diffusion generated designs for different tasks $\boldsymbol{c}$ that we consider in this work. The initial designs start off completely at random, and the optimized ones solve the task.

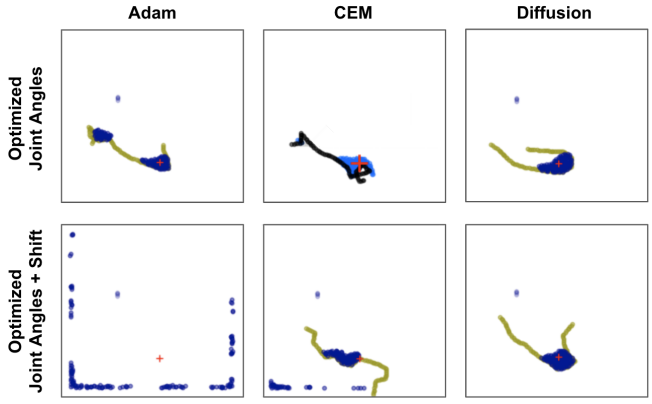

Figure 6. Examples of designs optimized with Adam, CEM, and guided diffusion using the generative model. Designs are initialized as random joint angles. Each design shown is the top scoring design for that optimizer, evaluated under the learned model, after having been trained on 1000 calls to the simulator (the limit of the x-axis in Figure 2).

where x_p^t are the coordinates of particle p after rolling out the simulation with the model f_M with initial conditons θ_{IC} and parameters θ_E. Note that the energy is evaluated on the last state of the simulation, hence ∇E needs to propagate through the whole simulation rollout.

In additon to the "Contain" environments described in section 3, we provide further example environments that we used for evaluation with the resulting designs from guided diffusion can be seen in Figure 5.

The task with the obstacle required that the design is fairly precise in order to bring the fluid into the cup, this is to highlight that the samples found from the diffusion model with conditional guidance + particle sampling in base distribution are able to precisely pinpoint these types of designs.

For the bi-modal task, where we have two possible minima of the cost function, we are able to capture both of the modes with the diffusion model.

In the case where we increase the dimensionality of the designs where we have x, y shift parameters for each of the tool joints, and the tools are disjoint, the diffusion model is able to come up with a parameterization that brings the particles in a desired configuration. However, the resulting designs are not robust and smooth, indicating that further modifications need to be made in form of constraints or regularization while guiding the reverse process to sample from $\tilde{\pi}(\boldsymbol{x})$.

D. Discussion on choice of guidance

As we will see in section 3, conditional guidance with corrected base distribution sampling tends to work better than energy guidance. In cases where the gradient of the energy function is expensive to evaluate, an energy-free alternative might be better, however this requires learning a conditional model, i.e. necessitates access to conditional samples.

Figure 7. Sampling (random search) from $p_1(\boldsymbol{x})$ and particle sampling in the bi-modal environment. We observe that even after increasing the number of samples, particle search further improves performance with same number of samples.

E. Energy guidance

We have found that using the gradient of the energy function as specified in equation (9) is a viable way of guiding the samples, albeit coming with the caveat of many expensive evaluations, see Figure 8. The guidance is very sensitive to the choice of the scaling factor λ, in our experiments we have found that a smaller scaling factor with many integration steps achieves better sample quality. Intuitively, this follows from the fact that it is difficult to guide 'bad' samples in the base distribution $p_1(\boldsymbol{x})$, which motivates the particle energy-based sampling scheme introduced in algorithm 1.

Further, we have looked at how to combine the gradient of the energy with the noised marginal score. We have found that re-scaling to have the same norm as the noised marginal improves the stability of guidance, as shown in *Figure 8*. Here, we have analyzed multiple functions with which we combined the energy gradient and the noised marginal score, we have looked at the following variants:

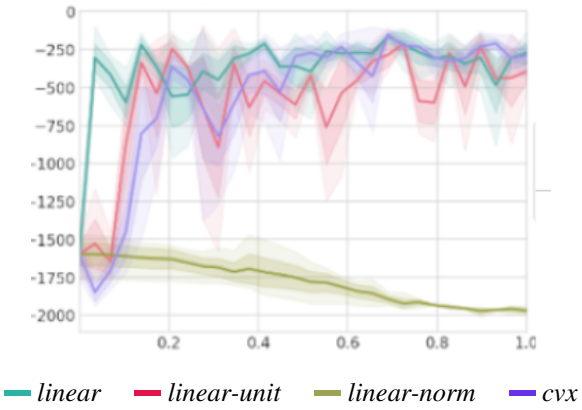

Figure 8. Performance of energy guidance depending on guidance scale λ (x axis) for different modifications to score of noised marginal.

- *linear* - simple linear addition of $\lambda \nabla E$.
- *linear-unit* - linear addition of $\lambda \frac{\nabla E}{\|\nabla E\|}$.
- *cvx* - convex combination of ∇E and ϵ_θ.
- *linear-norm* - linear addition of $\lambda \frac{\nabla E \|\epsilon_\theta\|}{\|\nabla E\|}$

	Contain	Ramp	With obstacle	Bi-modal	Multi-tool
Environment size	1x1	1x1	1x1	1x1	1x1
Rollout length	150	150	150	150	150
Initial fluid box(es)					
left	0.2	0.2	0.45	0.25, 0.65	0.2
right	0.3	0.3	0.55	0.35, 0.75	0.3
bottom	0.5	0.5	0.5	0.5	0.5
top	0.6	0.6	0.6	0.6	0.6
Reward sampling box					
left	0.4	0.8	0.2	0.25, 0.65	0.2
right	0.6	1.0	0.2	0.35, 0.75	0.3
bottom	0.1	0.0	0.2	0.1	0.2
top	0.3	0.2	0.2	0.2	0.5
Reward σ	0.1	0.1	0.05	0.1	0.1
# tools	1	1	1	1	16
# joint angles	16	16	16	16	1
Design parameters	joint angles, shift (optional)	joint angles	joint angles, shift	joint angles, shift	shift
Tool position (left)	[0.15, 0.35]	[0.15, 0.35]	[0.25, 0.35]	[0.3, 0.6]	[0.15–0.2, 0.35–0.4]
Tool Length	0.8	0.8	0.6	0.4	0.1
Additional obstacles	—	—	barrier halfway between cup and fluid, cup around goal	—	—

Table 1. Task Parameters.