Geometry-Informed Neural Networks

Arturs Berzins [*1,2] **Andreas Radler** [*1] **Sebastian Sanokowski** [1] **Sepp Hochreiter** [1,3] **Johannes Brandstetter** [1,3]

[*]Equal contribution
[1]LIT AI Lab, Institute for Machine Learning, JKU Linz, Austria
[2]SINTEF, Oslo, Norway
[3]NXAI GmbH, Linz, Austria
{berzins, radler, sanokowski, hochreit, brandstetter}@ml.jku.at

Abstract

Geometry is a ubiquitous language of computer graphics, design, and engineering. However, the lack of large shape datasets limits the application of state-of-the-art supervised learning methods and motivates the exploration of alternative learning strategies. To this end, we introduce geometry-informed neural networks (GINNs) to train shape generative models *without any data*. GINNs combine (i) learning under constraints, (ii) neural fields as a suitable representation, and (iii) generating diverse solutions to under-determined problems. We apply GINNs to several two and three-dimensional problems of increasing levels of complexity. Our results demonstrate the feasibility of training shape generative models in a data-free setting. This new paradigm opens several exciting research directions, expanding the application of generative models into domains where data is sparse.

1. Introduction

Geometry is a widely studied branch of mathematics, serving as a fundamental tool in various disciplines, including computer graphics, design, engineering, and physics. While in these fields, usually large datasets do not exist, important problems are often equipped with formal descriptions, such as objectives and constraints, opening the pathway for theory-informed learning.

Related attempts in theory-informed learning and neural optimization, most notably PINN (Raissi et al., 2019), have demonstrated that it is possible to train machine learning models using objectives and constraints alone, without relying on any data. However, an important difference is that problems in geometry are often under-determined and admit multiple solutions as exemplified by the variety of everyday and engineering objects.

In this work, we introduce *geometry-informed neural networks* (GINNs), formulated to produce shapes that conform to specified design constraints. By leveraging neural fields (Xie et al., 2022), GINNs offer detailed, smooth, and topologically flexible representations as closed level-sets, while being compact to store. Furthermore, to respect the inherent solution multiplicity we make GINNs *generative* using conditional neural fields. To address model collapse, we encourage diversity with an explicit loss. The overall concept and some experimental results on several different problems are showcased in Figure 1.

Practically, we first extend theory-informed learning with the generative aspect necessitated by under-determined problem settings. With this, we formalize the GINN paradigm, transforming a formal optimization problem into a tractable learning problem. Technical details cover enforcing and differentiating through constraints – especially connectedness –, facilitating diversity, impact of different architectures, defining metrics and problem scenarios, and scalibility towards 3D use cases.

2. Foundations

Theory-informed learning uses scientific knowledge to remove physically inconsistent solutions and reducing the variance of a model (Karpatne et al., 2017). Such knowledge can be included in the model via equations, logic rules, or human feedback (Dash et al., 2022; Muralidhar et al., 2018; Von Rueden et al., 2021). More formally, one searches for a solution f minimizing the objective $O(f)$ *s.t.* $f \in \mathcal{K}$, where $\mathcal{K}$ defines the feasible set in which the constraints are satisfied.

Neural fields (NFs) are NNs (typically multilayer-perceptrons (MLPs)) representing a function $f : x \mapsto y$ that maps a spatial and/or temporal coordinate x to a quantity y. Compared to discrete representations, NFs are significantly more memory-efficient while providing higher fidelity, as

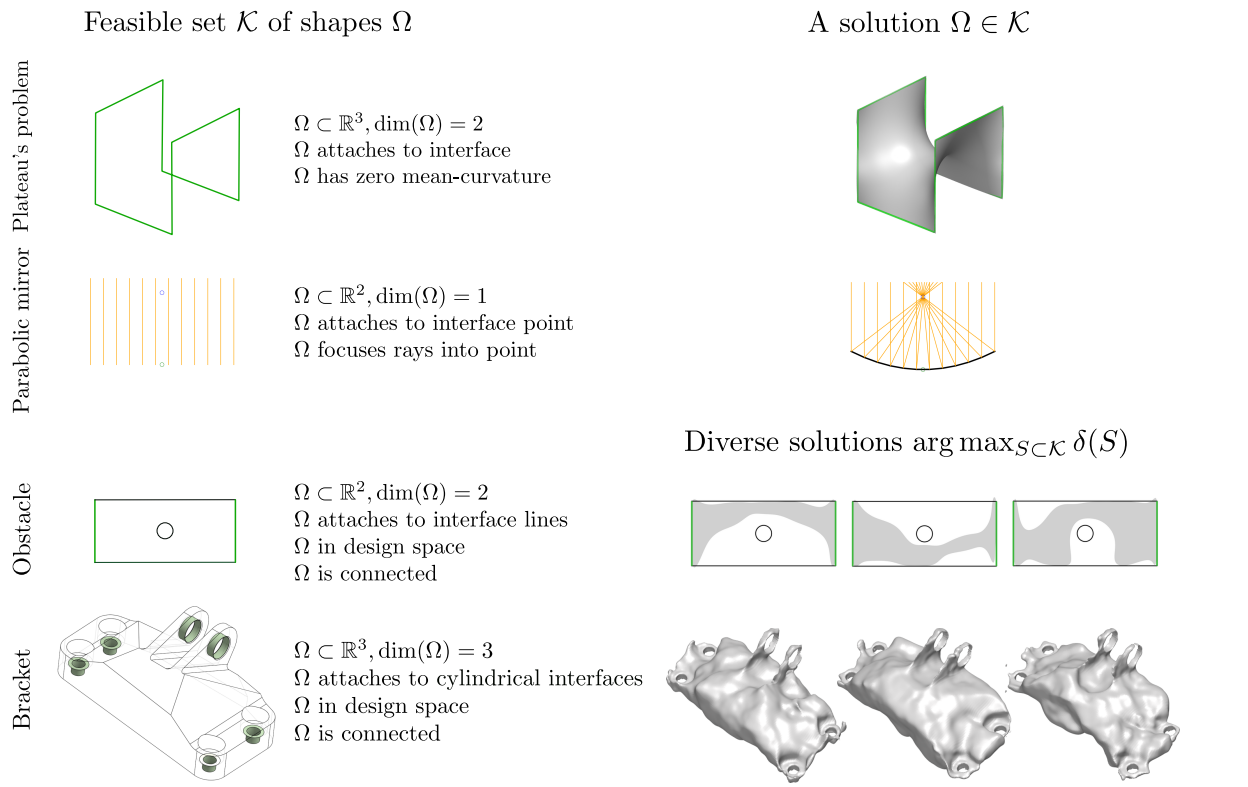


Figure 1: The GINN learning paradigm applied to four different geometry-constrained problems introduced in Section 4.2. A given set of constraints on the shape Ω defines the set of feasible shapes $\mathcal{K}$. A GINN is a neural network trained to find feasible shapes, which are unique in the top two rows. However, as often in geometry, the problems in the bottom two rows have multiple solutions. To produce diverse solutions $S \subset \mathcal{K}$ we maximise a diversity measure δ defined over the set of shapes S. Using only constraints and diversity, the GINN paradigm for shape generative modeling is entirely data-free.

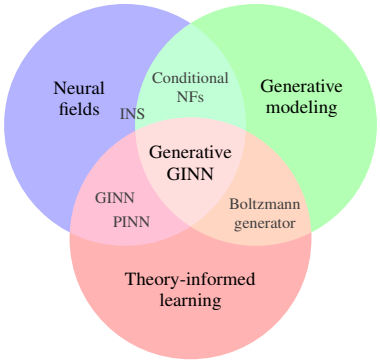


Figure 2: Generative GINNs lie at the intersection of neural fields, generative modeling, and theory-informed learning.

well as continuity and analytic differentiability (Xie et al., 2022). They have seen widespread success in various domains (Park et al., 2019; Mescheder et al., 2019; Mildenhall et al., 2021; Karras et al., 2021).

Implicit neural shapes (INSs) represent geometries through scalar neural fields. INSs also enjoy topological flexibility supporting shape reconstruction and generation. Several techniques exist to regularize and modify the inductive bias of prominent models (see Appendix D.2).

Deep generative modeling (Kingma & Welling, 2013; Goodfellow et al., 2014; Rezende & Mohamed, 2015; Tomczak, 2021) plays a central role in advancing deep learning and has enabled breakthroughs in various fields from natural language processing (Brown et al., 2020) to computer vision (Ho et al., 2020). Most related to our work are conditional NFs (see prior paragraph) and their applicability to deep generative design. *Conditional neural fields* encode multiple signals simultaneously by conditioning the NF on a latent variable z: $f(x) = F(x; z)$ where F is a base network, e.g. by concatenation (Park et al., 2019), hypernetworks (Ha et al., 2017), modulation (Mehta et al., 2021), or attention (Rebain et al., 2022).

Generative design refers to computational design methods, which can automatically conduct design exploration under constraints that are defined by designers (Jang et al., 2022). In contrast to generative modeling, its goal is not to mimic existing data, but to generate novel designs (Regenwetter et al., 2022; Shin et al., 2023).

3. Method

Consider an element f in some space $\mathcal{F}$. In this work, we focus on f being a function representing a geometry or a PDE solution. Let the set of constraints[1] $C(f) = [c_i(f)]$ be satisfied in the feasible set $\mathcal{K} = \{f \in \mathcal{F} | C(f) = 0\}$. Selecting the constraints C of a geometric nature lays the foundation for a *geometry-informed neural network* or *GINN*, which outputs a solution that satisfies the constraints: $f \in \mathcal{K}$.

3.1. Geometry-informed neural networks (GINNs)

Representation of a solution. While the GINN paradigm easily extends to other representations, we let a neural network $f : \mathcal{X} \mapsto \mathbb{R}, \mathcal{X} \subset \mathbb{R}^n$ approximate a signed-distance function . The sign of f *implicitly* defines the shape $\Omega = \{x \in \mathcal{X} | f(x) \leq 0\}$ and its boundary $\partial\Omega = \{x \in \mathcal{X} | f(x) = 0\}$, whereas the absolute value $|f(x)|$ shall approximate the distance of the point x to the closest boundary.

Constraints on a solution. The condition $f \in \mathcal{K}$ is effectively a hard constraint. We relax each constraint c_i into a differentiable loss $l_i : \mathcal{F} \mapsto [0, \infty)$ which describes the constraint violation. With the weights $\lambda_i > 0$, the total constrain violation of f is $L(f) = \sum_i \lambda_i l_i(f)$. This relaxes the constraint satisfaction problem $f \in \mathcal{K}$ into the unconstrained optimization problem $\min_f L(f)$. The constraints used in our experiments are collected in Table 1 and more are discussed in Table 5. By representing the set Ω through the function f, the geometric constraints on Ω (Tab. 1, col. 2) can be translated into functional constraints and differentiable losses on f (Tab. 1, col. 3 and col. 4).

Connectedness refers to an object Ω consisting of a single connected component. It is a ubiquitous feature enabling the propagation of mechanical forces, signals, energy, and other resources. Consequentially, enforcing connectedness is an important constraint for enabling GINNs. In Appendix C.3 we give more details on how we implement a connectedness loss on our INSs.

3.2. Generative GINNs

Representation of the solution set. The generator $G(z) = f$ maps a latent variable $z \in Z$ to a solution f. The solution set is hence the image of the latent set under the generator: $S = \mathrm{Im}_G(Z)$. Furthermore, the generator transforms the input probability distribution p_Z over Z to an output probability distribution p over S. In practice, the generator is a modulated base network producing a conditional neural field: $f(x) = F(x; z)$.

[1]For ease of notation, we transform inequality constraints to equality constraints.

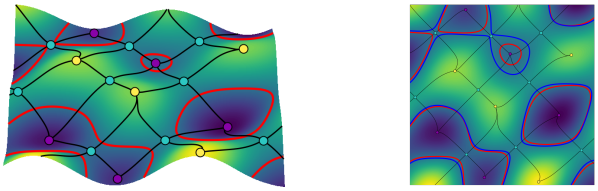


Figure 3: Our *connectedness loss* builds upon the *surface network*, in which integral paths (black) connect critical points. The key intuition behind our loss is that connected components (sub-level sets with boundaries in red) start at minima (purple) and connect via saddle points (turquoise). By penalizing values at specific saddle points, an update (right, blue) can connect components.

Constraints on the solution set. By adopting a probabilistic view, we extend the constraint violation to its expected value. This relaxes the relation $S \subseteq \mathcal{K}$ into $\min_S \mathcal{L}(S)$:

$$\int_S p(f) L(f) \, \mathrm{d}f = \mathop{\mathbb{E}}_{z \sim p_Z} [L(G(z))] = \mathcal{L}(S) \, . \qquad (1)$$

Diversity of the solution set. The last missing piece to training a generative GINN is making S a diverse collection of solutions. In the typical supervised generative modeling setting, the diversity of the generator is inherited from the diversity of the training dataset. The violation of this is studied under phenomena like *mode collapse* in GANs (Che et al., 2017). Exploration beyond the training data has been attempted by adding an explicit diversity loss, such as entropy (Noé et al., 2019), Coulomb repulsion (Unterthiner et al., 2018), determinantal point processes (Chen & Ahmed, 2020; Heyrani Nobari et al., 2021), pixel difference, and structural dissimilarity (Jang et al., 2022). We observe that simple generative GINN models are prone to mode-collapse, which we mitigate by adding a *diversity loss*. In essence, the diversity aggregates the pairwise dissimilarities of the elements of the set. As a shape dissimilarity, we use a modified function distance which can be related to the chamfer discrepancy (see Appendix E).

To summarize, **training a generative GINN corresponds to an unconstrained optimization problem** $\min_S \mathcal{L}(S) - \lambda_\delta \delta(S)$, where $\lambda_\delta > 0$ controls the potential trade-off between constraint violation $\mathcal{L}(S)$ and diversity $\delta(S)$ on the set $S = \mathrm{Im}_G(Z)$ of generated geometries.

3.3. Relation to PINNs

It has been observed that the fitting of INSs is related to PINN, e.g., via the eikonal equation (Gropp et al., 2020) or the Poisson problem (Sellán & Jacobson, 2023). We also observe empirically that many best practices for PINNs

	Set constraint $c(\Omega)$	Function constraint $c(f)$	Loss $l(f)$
Design region	$\Omega \subset \mathcal{E}$	$f(x) > 0 \; \forall x \notin \mathcal{E}$	$\int_{\mathcal{X}\setminus\mathcal{E}} [\min(0, f(x))]^2 \, \mathrm{d}x$
Interface	$\partial\Omega \supset \mathcal{I}$	$f(x) = 0 \; \forall x \in \mathcal{I}$	$\int_{\mathcal{I}} [f(x)]^2 \, \mathrm{d}x$
Prescribed normal	$n(x) = \bar{n}(x) \; \forall x \in \mathcal{I}$	$\frac{\nabla f(x)}{\|\|\nabla f(x)\|\|} = \bar{n}(x) \; \forall x \in \mathcal{I}$	$\int_{\mathcal{I}} \left[\frac{\nabla f(x)}{\|\|\nabla f(x)\|\|} - \bar{n}(x)\right]^2 \mathrm{d}x$
Mean curvature	$\kappa_H(x) = 0 \; \forall x \in \partial\Omega$	$\mathrm{div}\left(\frac{\nabla f(x)}{\|\|\nabla f(x)\|\|}\right) = 0 \; \forall x \in \partial\Omega$	$\int_{\partial\Omega} \left[\mathrm{div}\left(\frac{\nabla f(x)}{\|\|\nabla f(x)\|\|}\right)\right]^2 \mathrm{d}x$
Connectedness	See Figure 3 and Appendix C.3		

Table 1: Geometric constraints used in our experiments. The shape Ω and its boundary $\partial\Omega$ are represented implicitly by the (sub-)level set of the function f. If given, the shape must be contained within the *design region* $\mathcal{E} \subseteq \mathcal{X}$ and attach to the *interface* $\mathcal{I} \subset \mathcal{E}$ with a potentially prescribed *normal* $\bar{n}(x)$. n denotes the outward-facing normal and κ_H is the mean curvature, both of which can be computed from f in a closed form. More constraints are discussed in Table 5.

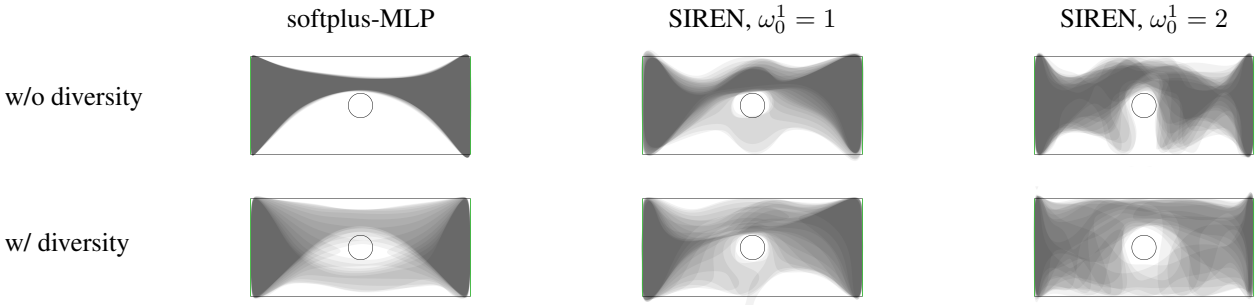


Figure 4: A superposition of 16 solutions found by different generative GINNs trained with and without a diversity loss. For the softplus-MLP, a diversity loss is needed to avoid mode-collapse. SIREN exhibits induced diversity, but adding the diversity loss further increases the diversity.

(Wang et al., 2023) transfer to GINNs. However the main differences are (1) PINNs primarily use differential, integral or fractional operators (2) GINNs typically require more loss terms than PINNs and (3) geometric problems are usually underdetermined. However, we find that the generative GINN paradigm can be transferred to under-determined physics systems as we demonstrate in Section 4.3.

4. Experiments

We experimentally demonstrate key aspects of GINNs (more details see Appendix A and B). To the best of our knowledge, data-free constraint-driven shape generative modeling is an unexplored field with no established baseline methods, problems, and metrics. In Appendix B.1, we define metrics for each constraint. We use these to compare different models and perform ablation studies in Appendices B.3 and B.2.

4.1. GINNs

Plateau's problem to demonstrate GINNs on a well-posed problem. Plateau's problem is to find the surface S with the minimal area given a prescribed boundary Γ (a closed curve in $\mathcal{X} \subset \mathbb{R}^3$). We train a GINN to find a shape under these constraints (for more details see Appendix A). Qualitatively, the result agrees with the known solution (see Figure 1).

Parabolic mirror to demonstrate a different geometric representation. Although we mainly focus on INSs, the GINN framework extends to other representations, such as explicit, parametric, or discrete shapes. Here, the GINN learns the height function of a mirror where all the reflected rays shall intersect in a single point (see Figure 1 and Appendix A).

4.2. Generative GINNs

Obstacle to introduce diversity and connectedness. We seek to find shapes that connect two interfaces in a rectangular domain and a round obstacle between them. The third row in Figure 1 depicts this set-up and three exemplary solutions, obtained with a generative GINN strategy since this problem admits infinitely many solutions.
In Table 3 and Figure 4 we perform and illustrate an ablation study. First, we observe that a conditioned MLP with a softplus activation (Dugas et al., 2000) trained *without a diversity loss shows mode-collapse* (Table 3, col. 2), while

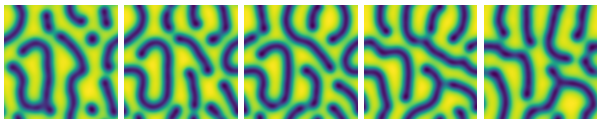


Figure 5: A generative PINN producing Turing patterns that morph during latent space interpolation. This is a result of searching for diverse solutions to an under-determined Gray-Scott system.

adding the diversity loss mitigates it (Table 3, col. 3). Alternatively, we observe that mode-collapse is also alleviated by switching to a model with a higher spectral bias (Tancik et al., 2020), such as a SIREN (Table 3, cols. 5, 7).

Jet engine bracket to demonstrate GINNs on a realistic 3D engineering design problem. The jet engine bracket is a challenging design problem. Here, we focus only on the geometric constraints: the shape must fit in a provided design space $\mathcal{E}$ and attach to five cylindrical interfaces $\mathcal{I}$ (Figure 1, row 4). In addition, we posit connectedness as a trivial requirement for structural integrity. The resulting shapes in Figure 1 demonstrate that the generative GINN can produce different shapes that closely satisfy the constraints (Table 4, col. 7). In the Appendix B.3, we provide further ablation studies for applied losses (e.g. diversity, connectedness and smoothness). Interestingly if we apply a smoothness loss on a SIREN network, also latent interpolation is improved (see Figure 10).

4.3. Generative PINNs

In physics, problems are often well-defined and have a unique solution. However, cases exist where the initial conditions are irrelevant and a non-particular PDE solution is sufficient, such as in chaotic systems or animations.
We demonstrate an analogous concept of *generative PINNs* on a *reaction-diffusion* system (first introduced by Turing (1952) to explain how patterns in nature, such as stripes, can form as a result of a simple physical process). Specifically we train a generative PINN to the Gray-Scott model (Pearson, 1993) visualized in Figure 5, further described in Appendix A. To the best of our knowledge, this is the first PINN that produces 2D Turing patterns in a data-free setting.

5. Conclusion

We have introduced geometry-informed neural networks demonstrating generative modeling driven solely by geometric constraints and diversity. After formulating the learning problem, we considered several constraints to define multiple problems of toy and realistic complexity. We solve these problems with GINNs demonstrating their viability and providing first insight into some of their key aspects.

Limitations and future work. Generative GINNs combine several known and novel components, each of which warrants an in-depth study of theoretical and practical aspects. It is worth exploring several alternatives to the shape dissimilarity and their aggregation into a diversity loss, architectures, and conditioning mechanism, as well as connectedness, whose current implementation is the computational bottleneck. An observed limitation of GINN training is the sensitivity to hyperparameters (see Appendix A.4) including the balancing of many losses, motivating the use of more advanced optimization techniques. In addition to scaling up the training, we believe tackling these aspects can help transfer the success of machine learning to practical applications in design synthesis and related tasks.

References

Atzmon, M. and Lipman, Y. SALD: sign agnostic learning with derivatives. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Ben-Shabat, Y., Hewa Koneputugodage, C., and Gould, S. DiGS: Divergence guided shape implicit neural representation for unoriented point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19323–19332, 2022.

Berzins, A., Ibing, M., and Kobbelt, L. Neural implicit shape editing using boundary sensitivity. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023.

Biasotti, S., De Floriani, L., Falcidieno, B., Frosini, P., Giorgi, D., Landi, C., Papaleo, L., and Spagnuolo, M. Describing shapes by geometrical-topological properties of real functions. *ACM Comput. Surv.*, 40(4), oct 2008a. ISSN 0360-0300.

Biasotti, S., Giorgi, D., Spagnuolo, M., and Falcidieno, B. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1):5–22, 2008b. ISSN 0304-3975. Computational Algebraic Geometry and Applications.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Brüel-Gabrielsson, R., Ganapathi-Subramanian, V., Skraba, P., and Guibas, L. J. Topology-aware surface reconstruction for point clouds. *Computer Graphics Forum*, 39(5): 197–207, 2020. doi: https://doi.org/10.1111/cgf.14079.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. Mode regularized generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Chen, W. and Ahmed, F. PaDGAN: Learning to Generate High-Quality Novel Designs. *Journal of Mechanical Design*, 143(3):031703, 11 2020. ISSN 1050-0472.

Chibane, J., Mir, A., and Pons-Moll, G. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020.

Clough, J. R., Byrne, N., Oksuz, I., Zimmer, V. A., Schnabel, J. A., and King, A. P. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(12):8766–8778, dec 2022. ISSN 1939-3539.

Dalmia, A. dalmia/siren, June 2020. URL https://doi.org/10.5281/zenodo.3902941.

Dash, T., Chitlangia, S., Ahuja, A., and Srinivasan, A. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1):1040, 2022.

Douglas, J. Solution of the problem of plateau. *Transactions of the American Mathematical Society*, 33(1):263–321, 1931. ISSN 00029947.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., and Garcia, R. Incorporating second-order functional knowledge for better option pricing. In Leen, T., Dietterich, T., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

Enflo, K. Measuring one-dimensional diversity. *Inquiry*, 0 (0):1–34, 2022.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231. AAAI Press, 1996.

Gabrielsson, R. B., Nelson, B. J., Dwaraknath, A., and Skraba, P. A topology layer for machine learning. In Chiappa, S. and Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1553–1563. PMLR, 26–28 Aug 2020.

Giampaolo, F., De Rosa, M., Qi, P., Izzo, S., and Cuomo, S. Physics-informed neural networks approach for 1d and 2d gray-scott systems. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):5, May 2022.

Goldman, R. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005. ISSN 0167-8396. Geometric Modelling and Differential Geometry.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Gropp, A., Yariv, L., Haim, N., Atzmon, M., and Lipman, Y. Implicit geometric regularization for learning shapes. In III, H. D. and Singh, A. (eds.), *Proceedings of Machine Learning and Systems 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3569–3579. PMLR, 13–18 Jul 2020.

Ha, D., Dai, A. M., and Le, Q. V. HyperNetworks. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, 2017.

Heyrani Nobari, A., Chen, W., and Ahmed, F. PcDGAN: A continuous conditional diverse generative adversarial network for inverse design. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 606–616, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hu, X., Li, F., Samaras, D., and Chen, C. Topology-preserving deep image segmentation. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jang, S., Yoo, S., and Kang, N. Generative design by reinforcement learning: Enhancing the diversity of topology optimization designs. *Computer-Aided Design*, 146: 103225, 2022. ISSN 0010-4485.

Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 852–863. Curran Associates, Inc., 2021.

Kiis, K., Wolfe, J., Wilson, G., Abbott, D., and Carter, W. Ge jet engine bracket challenge. https://grabcad.com/challenges/ge-jet-engine-bracket-challenge, 2013. Accessed: 2024-05-22.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kurochkin, S. V. Neural network with smooth activation functions and without bottlenecks is almost surely a morse function. *Computational Mathematics and Mathematical Physics*, 61(7):1162–1168, Jul 2021.

Leinster, T. and Cobbold, C. A. Measuring diversity: the importance of species similarity. *Ecology*, 93(3):477–489, March 2012.

Ma, B., Zhou, J., Liu, Y., and Han, Z. Towards better gradient consistency for neural signed distance functions via level set alignment. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17724–17734, Los Alamitos, CA, USA, jun 2023. IEEE Computer Society.

Marschner, Z., Sellán, S., Liu, H.-T. D., and Jacobson, A. Constructive solid geometry on neural signed distance fields. In *SIGGRAPH Asia 2023 Conference Papers*, SA '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703157.

McGough, J. S. and Riley, K. Pattern formation in the gray–scott model. *Nonlinear Analysis: Real World Applications*, 5(1):105–121, 2004. ISSN 1468-1218.

Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., and Chandraker, M. Modulated periodic activations for generalizable local functional representations. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14194–14203, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4460–4470, 2019.

Mezghanni, M., Boulkenafed, M., Lieutier, A., and Ovsjanikov, M. Physically-aware generative network for 3d shape modeling. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9326–9337, 2021. doi: 10.1109/CVPR46437.2021.00921.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Muralidhar, N., Islam, M. R., Marwah, M., Karpatne, A., and Ramakrishnan, N. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pp. 36–45. IEEE, 2018.

Nadimpalli, K. V., Chattopadhyay, A., and Rieck, B. A. Euler characteristic transform based topological loss for reconstructing 3d images from single 2d slices. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 571–579, 2023.

Nguyen, T., Pham, Q., Le, T., Pham, T., Ho, N., and Hua, B. Point-set distances for learning representations of 3d point clouds. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10458–10467, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

Novello, T., Schardong, G., Schirmer, L., da Silva, V., Lopes, H., and Velho, L. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Palmer, D., Smirnov, D., Wang, S., Chern, A., and Solomon, J. DeepCurrents: Learning implicit representations of shapes with boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019.

Parreño, F., Álvarez Valdés, R., and Martí, R. Measuring diversity. a review and an empirical analysis. *European Journal of Operational Research*, 289(2):515–532, 2021. ISSN 0377-2217.

Pearson, J. E. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for

solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Rana, S. (ed.). *Topological data structures for surfaces*. John Wiley & Sons, Chichester, England, March 2004.

Rebain, D., Matthews, M. J., Yi, K. M., Sharma, G., Lagun, D., and Tagliasacchi, A. Attention beats concatenation for conditioning neural fields. *Trans. Mach. Learn. Res.*, 2023, 2022.

Regenwetter, L., Nobari, A. H., and Ahmed, F. Deep Generative Models in Engineering Design: A Review. *Journal of Mechanical Design*, 144(7):071704, 03 2022. ISSN 1050-0472.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Sellán, S. and Jacobson, A. Neural stochastic poisson surface reconstruction. In *SIGGRAPH Asia 2023 Conference Papers*, SA '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703157.

Shin, S., Shin, D., and Kang, N. Topology optimization via machine learning and deep learning: a review. *Journal of Computational Design and Engineering*, 10(4):1736–1766, 07 2023. ISSN 2288-5048.

Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33: 7537–7547, 2020.

Tomczak, J. M. Why deep generative modeling? In *Deep Generative Modeling*, pp. 1–12. Springer, 2021.

Turing, A. A. M. The chemical basis of morphogenesis. *Philos. Trans. R. Soc. Lond.*, 237(641):37–72, August 1952.

Unterthiner, T., Nessler, B., Seward, C., Klambauer, G., Heusel, M., Ramsauer, H., and Hochreiter, S. Coulomb GANs: provably optimal nash equilibria via potential fields. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., et al. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2021.

Wang, F., Liu, H., Samaras, D., and Chen, C. TopoGAN: A topology-aware generative adversarial network. In *Proceedings of European Conference on Computer Vision*, 2020.

Wang, S. and Chern, A. Computing minimal surfaces with differential forms. *ACM Trans. Graph.*, 40(4):113:1–113:14, August 2021.

Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. An expert's guide to training physics-informed neural networks, 2023.

Westgaard, G. and Nowacki, H. Construction of Fair Surfaces Over Irregular Meshes . *Journal of Computing and Information Science in Engineering*, 1(4):376–384, 10 2001. ISSN 1530-9827. doi: 10.1115/1.1433484.

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. ISSN 1467-8659.

Yang, H., Sun, Y., Sundaramoorthi, G., and Yezzi, A. StEik: Stabilizing the optimization of neural signed distance functions and finer shape representation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A. Implementation and experimental details

We report additional details on the experiments and their implementation. We run all the experiments on a single GPU (one of NVIDIA RTX2080Ti, RTX3090, A40, or P40). The maximum GPU memory requirements are ca. 11GB for the jet engine bracket, ca. 7GB for the obstacle problem and less than a GB for the rest.

A.1. Neural network architectures

For the toy problems (parabolic mirror and Plateau's problem), we use very simple MLPs which we describe directly in the corresponding sections. In our main experiments, (obstacle and jet engine bracket), we use two more complex different MLP architectures described below.

Softplus-MLP. The neural network model f should be at least twice differentiable with respect to the inputs x, as necessitated by the computation of surface normals and curvatures. Since the second derivatives of an ReLU MLP is zero everywhere, we use the softplus activation function as a simple baseline. In addition, we add residual connections (Dugas et al., 2000) to mitigate the vanishing gradient problem and facilitate learning. We denote this architecture with "softplus-MLP".

SIREN. In some of our problem settings, early experiments indicated that the softplus-MLP cannot satisfy the given constraints. We therefore employ a SIREN network (Sitzmann et al., 2020) using the implementation of Dalmia (2020). As recommended, we tune ω_0^1, which controls the weight of the first layer at initialization and is largely responsible for the spectral properties of a SIREN model. As described by the authors, we find that important characteristics, such as expressivity and the latent space structure of a generative model, are highly sensitive to ω_0^1. For more detailed results, we refer to Section B.

A.2. Plateau's problem

Problem definition. Plateau's problem is to find the surface S with the minimal area given a prescribed boundary Γ (a closed curve in $\mathcal{X} \subset \mathbb{R}^3$). A *minimal surface* is known to have zero *mean curvature* κ_H everywhere. Minimal surfaces have boundaries and may contain intersections and branch points (Douglas, 1931) which cannot be represented implicitly. For simplicity, we select a suitable problem instance, noting that more appropriate geometric representations exist (Wang & Chern, 2021; Palmer et al., 2022). For an implicit surface, the mean curvature can be computed from the gradient and the Hessian matrix (Goldman, 2005). Altogether, we represent the surface as $S = \partial\Omega \cap \mathcal{X}$ and the two constraints are: $\Gamma \subset S$ and $\kappa_H(x) = 0 \; \forall x \in S$.

Model. The model is an MLP with $[3, 256, 256, 256, 1]$ neurons per layer and the $\tanh$ activation. We train with Adam (default parameters) for 10000 epochs with a learning rate of 10^{-3} taking around three minutes. The three losses (interface, mean curvature, and eikonal) are weighted equally but mean curvature loss is introduced only after 1000 epochs. To facilitate a higher level of detail, the corner points of the prescribed interface are weighted higher.

A.3. Parabolic mirror

Problem definition. For the parabolic mirror, the GINN learns the height function $f : [-1, 1] \mapsto \mathbb{R}$ of a mirror with the interface constraint $f(0) = 0$ and that all the reflected rays should intersect at the single point $(0, 1)$. The result in Figure 1 approximates the known solution: a parabolic mirror. This is a very basic example of caustics, an inverse problem in optics, which we hope inspires future work on analogous *vision-informed neural networks* leveraging the recent developments in neural rendering techniques.

Model. The model is an MLP with $[2, 40, 40, 1]$ neurons per layer and the $\tanh$ activation. We train with Adam (default parameters) for 3000 epochs with a learning rate of 10^{-3} taking around ten seconds.

A.4. Obstacle

Problem definition. Consider the domain $\mathcal{X} = [-1, 1] \times [-0.5, 0.5]$ and the design region that is a smaller rectangular domain with a circular obstacle in the middle: $\mathcal{E} = ([-0.9, 0.9] \times [-0.4, 0.4]) \setminus \{x_1^2 + x_2^2 \leq 0.1^2\}$. There is an interface consisting of two vertical line segments $\mathcal{I} = \{(\pm 0.9, x_2) | -0.4 \leq x_2 \leq 0.4\}$ with the prescribed outward facing normals $\bar{n}(\pm 0.9, -0.4 \leq x_2 \leq 0.4) = (\pm 1, 0)$.

Conditioning the model. For training the conditional models, we approximate the one-dimensional latent set $Z = [-1, 1]$ with $N = 16$ fixed equally spaced samples. This enables the reuse of some calculations across epochs and results in a well-structured latent space, illustrated through latent space interpolation in Figure 4.

Hyperparameter tuning. The obstacle experiment serves as a proof of concept for including and balancing several losses, in particular the connectedness loss. The models are a softplus-MLP and a SIREN network with $\omega_0^1 \in \{1, 2\}$. We train with Adam (default settings) and the hyperparameters in Table 2. Leveraging the similarity to PINNs, we follow many practical suggestions discussed in Wang et al. (2023). We find that a good strategy for loss balancing is to start with the *local* losses (i.e. they are only applied in a subset of the domain $\mathcal{X}$, mainly interface, envelope, normal losses) and then incorporate *global* losses (eikonal, connectedness, smoothness losses). In general, we observe that the λs of the global loss should be kept lower than those of the local

losses in order not to destroy the local shape structure. By adding one loss at a time, we binary-search an appropriate weight while preserving the overall balance. After having found some reasonably good hyperparameters for the λs, we recommend tuning ω_0^1 as it is crucial to control the smoothness of the shape. After having tuned ω_0^1, the λs may be tuned for further improvements.

Computational cost. The total training time is around an hour for the GINN (single shape) and 5 hours for the generative GINN (trained on 16 shapes). The bulk of the computation time (often more than 90%) is taken by the connectedness loss. To alleviate this, we recompute the critical points every 10 epochs and use the previous points as a warm start. While this works well for the softplus-MLP, it does not work reliably for SIREN networks since the behavior of their critical points is more spurious. This presents an avenue for future improvement.

A.5. Jet engine bracket

Problem definition. The problem specification draws inspiration from an engineering design competition hosted by General Electric and GrabCAD (Kiis et al., 2013). The challenge was to design the lightest possible lifting bracket for a jet engine subject to both physical and geometrical constraints. Here, we focus only on the geometric constraints: the shape must fit in a provided design space $\mathcal{E}$ and attach to five cylindrical interfaces $\mathcal{I}$ (Figure 1, row 4). In addition, we posit connectedness as a trivial requirement for structural integrity.

Hyperparameters. The jet engine bracket (JEB) is our most complex experiment. In contrast to the obstacle experiment, we only SIREN worked. In addition, we increase the sampling density around the interfaces. We train with Adam (default settings) and the hyperparameters summarized in Table 2. The total training time is around 17 hours for the GINN (single shape) and 26 hours for the generative GINN (trained on 4 shapes).

Conditioning the model. In the generative GINN setting, we condition SIREN using input concatenation which can be interpreted as using different biases at the first layer. As we refer in the main text, we leave more sophisticated conditioning techniques for future work. We use $N = 4$ different *fixed* latent codes spaced equally in $Z = [-0.1, 0.1]$.

Tuning ω_0^1. We tune ω_0^1 and find that 8.0 leads to satisfying shapes, while the values 6.5 (see Figure 8(b) and Table 4, col. 3) and 10.0 produced shapes that were too smooth or wavy, respectively.

Spatial resolution. The curse of dimensionality implies that with higher dimensions, exponentially (in the number of dimensions) more points are needed to cover the space equidistantly. Therefore, in 3D, substantially more points (and consequently memory and compute) are needed than in 2D. In our experiments, we observe that a low spatial resolution around the interfaces prevents the model from learning high-frequency details, likely due to a stochastic gradient. Increased spatial resolution results in a better learning signal and the model picks up the details easier. For memory and compute, we increase the resolution much more around the interfaces and less so elsewhere.

A.6. Reaction-diffusion

Problem definition. Having developed a generative GINN that is capable of producing diverse solutions to an underdetermined problem, we ask if this idea generalizes to other areas. In physics, problems are often well-defined and have a unique solution. However, cases exist where the initial conditions are irrelevant and a non-particular PDE solution is sufficient, such as in chaotic systems or animations.
We conclude the experimental section by demonstrating an analogous concept of *generative PINNs* on a *reaction-diffusion* system. A prominent model of such a system is the Gray-Scott model (Pearson, 1993), which produces a variety of patterns by changing just two parameters – the feed-rate α and the kill-rate β – in the following PDE:

$$\begin{aligned} \frac{\partial u}{\partial t} &= D_u \Delta u - uv^2 + \alpha(1-u) \\ \frac{\partial v}{\partial t} &= D_v \Delta v + uv^2 - (\alpha+\beta)v \end{aligned} \tag{2}$$

This PDE describes the concentration u, v of two substances U, V undergoing the chemical reaction $U + 2V \to 3V$. The rate of this reaction is described by uv^2, while the rate of adding U and removing V is controlled by the parameters α and β. Crucially, both substances undergo diffusion (controlled by the coefficients D_u, D_v) which produces an instability leading to rich patterns around the bifurcation line $\alpha = 4(\alpha + \beta)^2$.
Computationally, these patterns are typically obtained by evolving a given initial condition $u(x, t = 0) = u_0(x)$, $v(x, t = 0) = v_0(x)$ on some domain with periodic boundary conditions. A variety of numerical solvers can be applied, but previous PINN attempts fail without data (Giampaolo et al., 2022). To demonstrate a generative PINN on a problem that admits multiple solutions, we omit the initial condition and instead consider stationary solutions, which are known to exist for some parameters α, β (McGough & Riley, 2004). We use the corresponding stationary PDE ($\partial u/\partial t = \partial v/\partial t = 0$) to formulate the residual losses:

$$\begin{aligned} L_u &= \int_{\mathcal{D}} (D_u \Delta u - uv^2 + \alpha(1-u))^2 \,\mathrm{d}x \\ L_v &= \int_{\mathcal{D}} (D_v \Delta v + uv^2 - (\alpha+\beta)v)^2 \,\mathrm{d}x \end{aligned} \tag{3}$$

To avoid trivial (i.e. uniform) solutions, we encourage non-zero gradient with a loss term $-\max(1, \int_{\mathcal{D}} (\nabla u(x))^2 +$

Hyperparameter	Obstacle (2D)		JEB (3D)
Architecture	Residual-MLP	SIREN	SIREN
Layers	$[3, 4 \times 512, 1]$	$[3, 4 \times 64, 1]$	$[4, 5 \times 256, 1]$
Activation	softplus	sine	sine
ω_0 of first layer for SIREN	n/a	[1.0, 2.0]	8.0
Learning rate	0.001	0.001	0.001
Learning rate schedule	$0.5^{t/1000}$	$0.5^{t/1000}$	
Iterations	3000	3000	5000
$\lambda_{\text{interface}}$	1	1	1
$\lambda_{\text{envelope}}$	1	1	10^{-1}
$\lambda_{\text{obstacle}}$	10^{-1}	1	n/a
λ_{normal}	10^{-2}	10^{-2}	10^{-6}
λ_{eikonal}	10^{-5}	10^{-5}	10^{-9}
$\lambda_{\text{connectedness}}$	10^{-5}	10^{-2}	10^{-2}
$\lambda_{\text{diversity}}$	10^{-5} to 10^{-4}	10^{-2}	10^{-5} to 10^{-3}
$\lambda_{\text{smoothness}}$	n/a	n/a	10^{-8} to 10^{-7}

Table 2: Hyperparameters for the generative 2D obstacle and 3D jet engine bracket experiments. The input is a 2D or 3D point concatenated with a 1D latent vector. For both experiments, the initial learning rate is halved every 1000 iterations. In the layers description e.g. 512x4 means that there were 4 layers of 512 width. Interestingly, the SIREN network overall had fewer parameters, while fitting a more complex shape.

$(\nabla v(x))^2 \, \mathrm{d}x)$. Similar to the 3D geometry experiment, we find that architecture and initialization are critical. Using the diffusion coefficients $D_v = 1.2 \times 10^{-5}$, $D_u = 2D_v$ and the feed and kill-rates $\alpha = 0.028, \beta = 0.057$, the generative PINN produces diverse and smoothly changing pattern of worms, illustrated in Figure 5. To the best of our knowledge, this is the first PINN that produces 2D Turing patterns in a data-free setting.

Model. We use two identical SIREN networks for each of the fields u and v. They have two hidden layers of widths 256 and 128. We enforce periodic boundary conditions on the unit domain $\mathcal{X} = [0, 1]^2$ through the encoding $x_i \mapsto (\sin 2\pi x_i, \cos 2\pi x_i)$ for $i = 1, 2$. With this encoding, we use $\omega_0 = 3.0$ to initialize SIREN. We also find that the same shaped Fourier-feature network (Tancik et al., 2020) with an appropriate initialization of $\sigma = 3$ works equally well.

Training. We compute the gradients and the Laplacian using finite differences on a 64×64 grid, which is randomly translated in each epoch. Automatic differentiation produces the same results for an appropriate initialization scheme, but finite differences are an order of magnitude faster. The trained fields u, v can be sampled at an arbitrarily high resolution without displaying any artifacts.
We use the loss weights $\lambda_{\text{residual}} = 1$, $\lambda_{\text{gradient}} = 10^{-4}$, and $\lambda_\delta = 10^{-7}$. The generative PINNs are trained with Adam for 20000 epochs with a 10^{-3} learning rate taking a few minutes.

B. Evaluation

B.1. Metrics

We introduce several metrics for each individual constraint independently. Let $vol(P) = \int_P dP$ be the generalized volume of P. We will use the chamfer discrepancy (Nguyen et al., 2021) to compute the dissimilarity between two shapes P and Q. For better interpretability, we take the square root of the common definition of chamfer discrepancy

$$CD_1(P, Q) = \sqrt{\frac{1}{|Q|} \sum_{x \in Q} \min_{y \in P} ||x - y||_2^2} \tag{4}$$

and, similary, for the two-sided chamfer discrepancy

$$CD_2(P, Q) = \sqrt{CD_1(P, Q)^2 + CD_1(Q, P)^2} \tag{5}$$

Reusing the notation from the paper, let $\mathcal{E}$ be the design region, $\delta\mathcal{E}$ the boundary of the design region, $\mathcal{I}$ the interface consisting of $n_{\mathcal{I}}$ connected components, $\mathcal{X}$ the domain, Ω the shape and $\delta\Omega$ its boundary.

Shape in design region. We introduce two metrics to quantify how well a shape fits the design region. Intuitively for

3D, the first metric quantifies how much volume is outside the design region $\mathcal{E}$ compared to the overall volume that is available. The second metric compares how much surface area intersects the boundary of the design region.

- $\frac{vol(\Omega \setminus \mathcal{E})}{vol(\mathcal{X} \setminus \mathcal{E})}$: The d-volume (i.e. volume for $d = 3$ or area for $d = 2$) outside the design region, divided by the total d-volume outside the design region.
- $\frac{vol(\Omega \cap \delta\mathcal{E})}{vol(\delta\mathcal{E})}$: The $(d-1)$-volume (i.e. the surface area for $d = 3$ or length of contours for $d = 2$) of the shape intersected with the design region boundary, normalized by the total $(d-1)$-volume of the design region.

Fit to the interface. To measure the goodness of fit to the interface, we use the *one-sided* chamfer distance of the boundary of the shape to the interface, as we do not care if some parts of the shape boundary are far away from the interface, as long as there are some parts of the shape which *are* close to the interface. A good fit is indicated by a 0 value.

- $CD_1(\Omega, \mathcal{I})$: The average minimal distance from sampled points of the interface to the shape boundary.

Connectedness. For the connectedness, we care whether the shape and whether the interfaces are connected. Since it is possible that the shape connects though paths that are outside the design region, we also introduce a metric that excludes such parts. The function $DC(\Omega)$ denotes all connected components of a shape Ω except the largest. We define the metrics as follows:

- $b_0(\Omega)$: The zeroth Betti number represents the number of connected components of the shape. The target in our work is always 1.
- $b_0(\Omega \cap \mathcal{E})$: The zeroth Betti number of the shape restricted to the design region.
- $\frac{vol(DC(\Omega))}{vol(\mathcal{E})}$: To measure the d-volume (i.e. volume for $d = 3$ and area for $d = 2$) of disconnected components, we compute their volume and normalize it by the volume of the design region.
- $\frac{vol(DC(\Omega \cap \mathcal{E}))}{vol(\mathcal{E})}$: Measures the d-volume of disconnected components *inside the design region*.
- $\frac{CI(\Omega, \mathcal{I})}{n_{\mathcal{I}}}$ computes the share of connected interfaces. If an interface is an ϵ-distance from a connected component of a shape, we consider it connected to the shape. This metric then represents the maximum number of connected interfaces of any connected component, divided by the total number of interface components. By default, we set $\epsilon = 0.01$ when then domain bounds are comparable to the unit cube.

Diversity. We define the diversity δ_{mean} on a finite set of shapes $S = \{\Omega_i, i \in [N]\}$ as follows:

$$\delta_{\text{mean}}(S) = \left[\frac{1}{N} \sum_{i \in [N]} \left(\frac{1}{N-1} \sum_{j \neq i \in [N]} CD_2(\Omega_i, \Omega_j) \right)^{\frac{1}{2}} \right]^2 . \tag{6}$$

Smoothness. There are many choices of smoothness measures in multiple dimensions. In this paper, we use a Monte Carlo estimate of the *surface strain* (Goldman, 2005) (also mentioned in Table 1). To make the metric more robust to large outliers (e.g. tiny disconnected components have very large curvature and surface strain), we clip the surface strain of a sampled point $x_i, i \in [N]$ with a value $\kappa_{\text{max}} = 1000$.

$$E_{\text{strain}}(\Omega) = \frac{1}{N} \sum_{i \in [N]} \min \left[\text{div}^2 \left(\nabla \frac{f(x_i)}{|f(x)|} \right), \kappa_{\text{max}} \right] \tag{7}$$

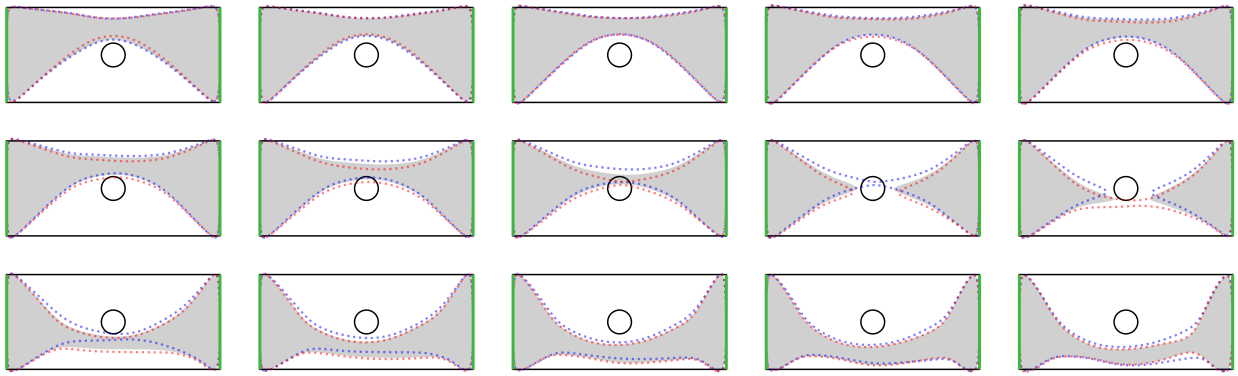
B.2. Obstacle

We perform quantitative evaluations of different configurations of hyperparameters on the obstacle problem. The results can be found in Table 3. In the following, we summarize the main findings.

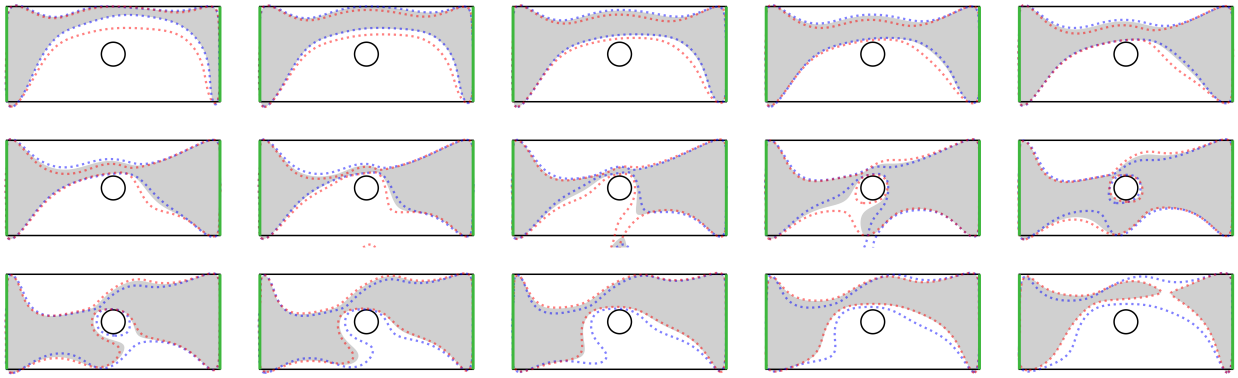
SIREN is more expressive than softplus-MLPs. While both types of models (SIRENs and softplus-MLPs) are able to solve the task, a big difference is visible in the diversity. A SIREN without explicit diversity loss beats the softplus-MLP by an order of magnitude. This suggests that SIREN has an inductive bias that promotes diversity.

Explicit diversity loss promotes higher diversity. Using an explicit diversity loss improves the diversity δ_{mean} across all experiments (cf. column 3 vs. 2, 5 vs. 4 and 7 vs. 6). An ablation of the diversity loss for softplus-MLP results in mode collapse as shown in Figure 4.

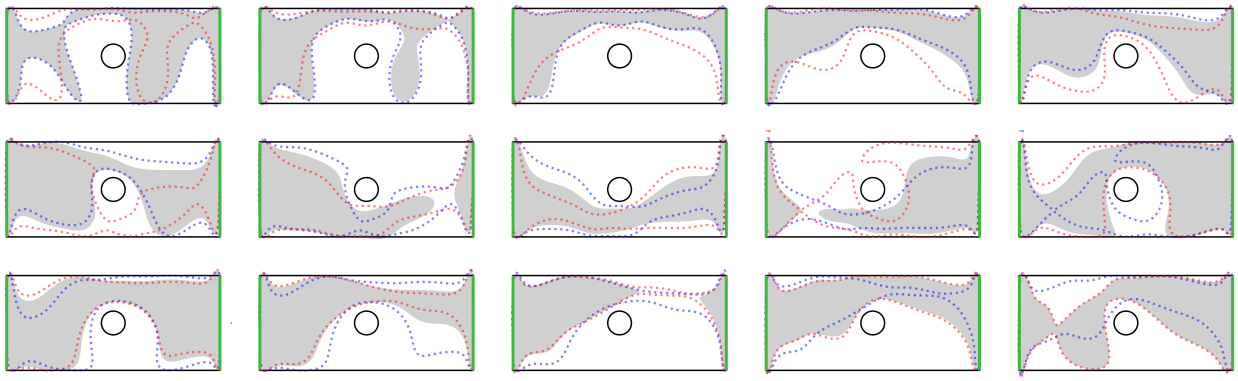
Interpolation degrades with higher spectral bias. An important property of a generative model is a structured latent space, which is key to sample similar outputs, perform interpolation, exploration, and generalize. We explore the interpolation property on the different models. As the models are trained on 16 equidistant fixed latents, the interpolation is performed on the 15 corresponding mid-points. In Figure 6, we compare a softplus-MLP (a) and SIREN with $\omega_0^1 = 1.0$ (b) and $\omega_0^1 = 2.0$ (c). Generally, we observe that the interpolation quality degrades from the softplus-MLP to SIREN with $\omega_0^1 = 1.0$ to a SIREN with $\omega_0^1 = 2.0$.



(a)



(b)



(c)

Figure 6: Interpolations of (a) softplus-MLP, (b) a SIREN with $\omega_0^1 = 1.0$, and (c) a SIREN with $\omega_0^1 = 2.0$. The gray shapes are generated with the latent mid-point of the shapes marked with the red and blue dots. The interpolation degrades with higher ω_0^1, e.g., in row 3, col. 3 in (c), the interpolated shape is disconnected while both neighboring shapes are connected.

	softplus-MLP		SIREN			
Figures	4, 6(a)	4	4, 6(b)	4	1, 4, 6(c)	4
Model						
ω_0^1	-	-	1	1	2	2
Loss						
λ_{div}		0		0		0
Metrics for a single shape Ω						
Connectedness						
$\downarrow b_0(\Omega)$	1.13	1.00	1.12	1.00	1.06	1.00
$\downarrow b_0(\Omega \cap \mathcal{E})$	1.13	1.00	1.06	1.00	1.00	1.00
$\downarrow \frac{vol(DC(\Omega))}{vol(\mathcal{E})}$	0.018	0.0	$6.4e{-3}$	0.0	$1.0e{-5}$	0.0
$\downarrow \frac{vol(DC(\Omega \cap \mathcal{E}))}{vol(\mathcal{E})}$	0.025	0.0	$8.9e{-3}$	0.0	0.0	0.0
$\uparrow \frac{CI(\Omega, \mathcal{I})}{n_{\mathcal{I}}}$	1.80	2.00	1.91	2.00	2.00	2.00
Interface						
$\downarrow CD_1(\Omega, \mathcal{I})$	$3.0e{-3}$	$3.8e{-5}$	$2.3e{-3}$	$2.3e-3$	$1.6e{-3}$	$2.2e{-3}$
Design region						
$\downarrow \frac{vol(\Omega \cap \delta\mathcal{E})}{vol(\delta\mathcal{E})}$	0.037	0.095	0.13	0.13	0.11	0.045
$\downarrow \frac{vol(\Omega \setminus \mathcal{E})}{vol(\mathcal{X} \setminus \mathcal{E})}$	$3.7e{-3}$	$3.8e{-3}$	$9.4e{-3}$	0.010	$5.9e{-3}$	$3.7e{-3}$
Metrics for shapes $S = \{\Omega_i\}$						
Diversity						
$\uparrow \delta_{\text{mean}}$	0.12	0.0076	0.1	0.067	0.14	0.073

Table 3: Metrics for different models trained on the obstacle problem. Some cells are left empty for better visual interpretability. In this case the default hyperparameter of Table 2 was taken. All models were generative GINNs, trained on 16 shapes. The metrics for a single shape were averaged across all 16 shapes.

B.3. Jet engine bracket

We show the results of some model variants and ablations trained in Table 4. The default setups (as reported in Table 2) correspond to the columns 2 and 7.

Smoothness regularization Figure 7 shows several shapes produced by a SIREN model. While these closely satisfy the constraints (Table 4, col. 5), they exhibit undulations (high surface waviness) due to the high-frequency bias of the model. We find that controlling the initialization can counteract this, but also interferes with the constraint satisfaction (Figure 8(b), col. 3). Instead, this can be controlled with an additional smoothness regularization term. Many possible fairing energies exist, each leading to different surface qualities (Westgaard & Nowacki, 2001), but we penalize the *surface strain*: $\int_{\partial\Omega\setminus\mathcal{I}} \kappa_1^2(x) + \kappa_2^2(x)\,\mathrm{d}x$, where κ_1 and κ_2 are the principal curvatures. The resulting shapes in Figure 1 demonstrate that the generative GINN can produce different shapes that closely satisfy the constraints (Table 4, col. 7). The smoothness regularization also helps structure the latent space aiding interpolation, i.e. generalization (Figure 10).

Sensitivity to ω_0^1. Column 3 and Figure 8(b) indicates that the interface fit $CD_1(\Omega, \mathcal{I})$ is worse by several orders of magnitude compared to the baseline setting. This is explained by the lower ω_0^1 leading to a smoother shape which in turn leads to a worse fit of the interfaces. As also observed previously, SIREN is highly sensitive to the ω_0^1 parameter.

Connectedness loss is crucial for connected shapes. Column 4 and Figure 8(a) ablate the connectedness loss. Qualitatively, this leads to a spurious shape. Quantitatively, the zeroth Betti number $b_0(\Omega)$ (similary, $b_0(\Omega \cap \mathcal{E})$) is very high, i.e., there are many disconnected components. Fur-

	GINN					Generative GINN		
Figure		8(b)	8(a)	7	9	1		
Model								
num_shapes	1	1	1	1	1	4	4	4
ω_0^1	8	6.5	8	8	8	8	8	8
Losses								
$\lambda_{\text{connectedness}}$			0					
$\lambda_{\text{smoothness}}$				0				
λ_{normal}					0			
λ_{div}							0	
λ_{eikonal}								0
Metrics for Ω								
Connectedness								
$\downarrow b_0(\Omega)$	4	1	33	5	10	4.00	8.75	4.75
$\downarrow b_0(\Omega \cap \mathcal{E})$	1	1	27	3	3	2.50	2.00	2.00
$\downarrow \frac{vol(DC(\Omega))}{vol(\mathcal{E})}$	$3.5e{-7}$	0	$1.2e{-2}$	$2.5e{-5}$	$1.2e{-5}$	$4.7e{-5}$	$2.4e{-5}$	$9.0e{-6}$
$\downarrow \frac{vol(DC(\Omega \cap \mathcal{E}))}{vol(\mathcal{E})}$	0	0	$3.2e{-2}$	$6.4e{-5}$	$2.9e{-5}$	$1.0e{-4}$	$4.5e{-5}$	$2.0e{-5}$
$\uparrow \frac{CI(\Omega,\mathcal{I})}{n_{\mathcal{I}}}$	1.00	1.00	0.17	1.00	1.00	1.00	1.00	1.00
Interface								
$\downarrow CD_1(\Omega, \mathcal{I})$	$6.6e{-4}$	$1.1e{-2}$	$9.2e{-4}$	$9.6e{-4}$	$7.8e{-4}$	$1.9e{-3}$	$1.3e{-3}$	$1.1e{-3}$
Design region								
$\downarrow \frac{vol(\Omega \cap \delta\mathcal{E})}{vol(\delta\mathcal{E})}$	$7.4e{-5}$	$3.1e{-3}$	$1.3e{-4}$	$7.8e{-5}$	$1.5e{-4}$	$2.0e{-4}$	$1.2e{-4}$	$9.2e{-5}$
$\downarrow \frac{vol(\Omega \setminus \mathcal{E})}{vol(\mathcal{X} \setminus \mathcal{E})}$	$3.3e{-5}$	$4.6e{-3}$	$6.5e{-5}$	$5.8e{-5}$	$6.5e{-5}$	$2.3e{-4}$	$1.7e{-4}$	$1.1e{-4}$
Smoothness								
$\downarrow E_{\text{strain}}(\Omega)$	182.7	248.5	636.7	401.8	181.0	211.9	245.2	237.5
Metrics for S								
Diversity								
$\uparrow \delta_{\text{mean}}(S)$						0.061	0.034	0.033

Table 4: Metrics for the jet engine bracket problem. All models used the SIREN architecture. The default settings are in the second (for GINN) and sixth column (for generative GINN). For generative GINNs, the metrics for a single shape were computed by taking the mean of the 4 generated shapes. Note that for the GINN experiments the diversity metric is omitted as it is only well-defined on a set of shapes $S = \{\Omega_i\}$.

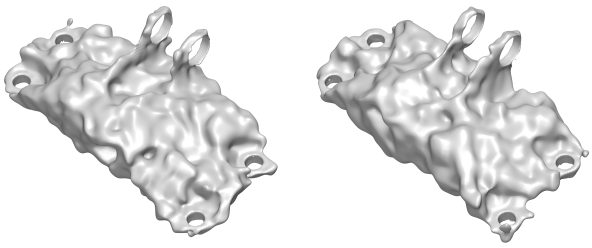


Figure 7: Generative GINN trained without *the smoothness loss*, contrasting the smoother shapes in Figure 1. These shapes satisfy the constraints well but display high surface *undulation* (waviness) due to SIREN's bias toward high-frequencies.

thermore, the share of connected interfaces $\frac{CI(\Omega,\mathcal{I})}{n_{\mathcal{I}}}$ is only 0.17. Since for this problem there are 6 interfaces to connect, a value of 0.17 implies that none of the interfaces are connected, indicating the importance of the connectedness loss.

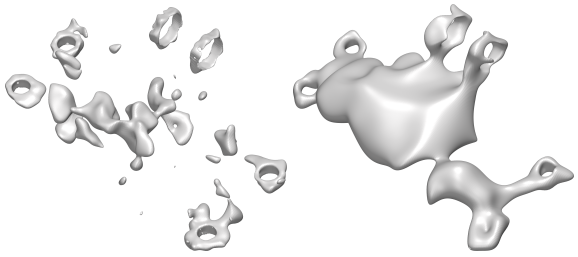
Normal loss facilitates learning at the interfaces. Column 6 and Figure 9 ablate the normal loss. This leads to similar interface metrics, but the connectedness metrics are worse, implying that there might be small disconnected components at the interface.

Explicit diversity loss and eikonal loss improve diversity. Comparing Table 4, col. 7 to col. 8 shows that not using the diversity loss halves the diversity $\delta_{\text{mean}}(S)$. Interestingly, also not using the eikonal loss reduces the diversity. We hypothesize, that the reason is that for training we compute a diversity loss on neural fields, sampled at points close to the individual boundaries. In contrast, the diversity metric (defined in section B.1) is computed using shapes *at the zero level set* of those fields with the chamfer-discrepancy as a pseudo-distance measure. Using the eikonal loss, leads to enforcing a more regular neural field, which in turn makes the diversity on neural fields more suitable.

Interpolation improves with smoothing. Figure 10 shows interpolations of models trained with and without the smoothness loss. The bottom row indicates that the conditional SIREN models do not form a strong latent space structure, and therefore does not allow for meaningful interpolation. Surprisingly, the application of the smoothness loss (top row) mitigates this. Understanding the precise mechanism behind this is left for future work.

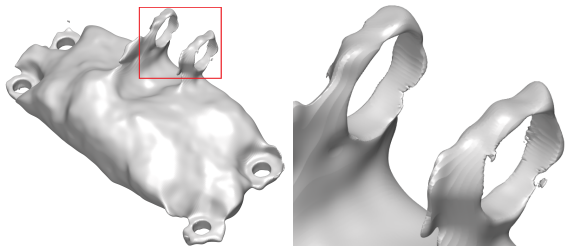
C. Connectedness

We provide additional details on our approach to the connectedness loss. We start with a brief overview and then detail the two major steps.



(a) (b)

Figure 8: Ablation of (a) connectedness, (b) initialization scheme. (a) A shape generated by a GINN using SIREN trained *without the connectedness loss*. The shape fits the design space and interfaces well, but it consists of many spurious disconnected components failing to connect the prescribed boundaries. (b) A shape generated by a GINN using SIREN with a *poor initialization* of $\omega_0 = 6.5$ for the first layer. The shape is too smooth and does not fit the interfaces well.



(a) (b)

Figure 9: Ablation of the normal loss at the interfaces. The close-up in (b) shows the upper interfaces where there are small disconnected components.

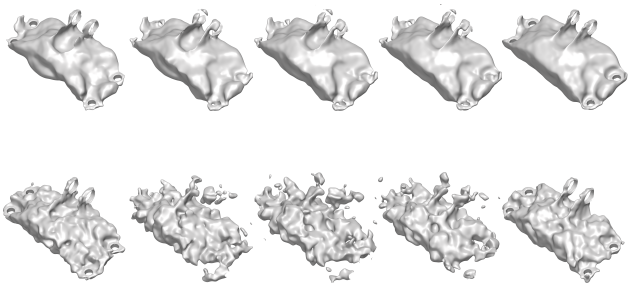


Figure 10: Latent space interpolation for models trained with and without smoothness regularization. For each row, the left and right shapes correspond to two out of four fixed latent codes used during training. The middle shapes are generated by linearly interpolating these two latent codes. Without the smoothness loss, SIREN leads to wavy shapes with poor latent space structure. In contrast, the smoothness loss helps structure the latent space and leads to more desirable results during the interpolation.

C.1. High-level overview.

In the context of machine learning, connectedness constraints have been multiply applied in segmentation (Wang et al., 2020; Clough et al., 2022; Hu et al., 2019), surface reconstruction (Brüel-Gabrielsson et al., 2020), and 3D shape generation with voxels (Nadimpalli et al., 2023), point-clouds (Gabrielsson et al., 2020) and INSs (Mezghanni et al., 2021).

Despite connectedness and other topological properties being discrete-valued, *persistent homology* (PH) has been the main tool allowing the formulation of a differentiable loss. In brief, it identifies topological features (like connected components or holes) and quantifies their *persistence*, matching the birth and death of each feature to a pair of points, whose values can then be adjusted to achieve the desired topological properties. However, all previous works compute PH from a cell complex, meaning the continuous function, such as the INS, if first discretized into a real-valued cubical complex.

We implement an alternative approach, in which we locate the birth and death pairs from the continuous function through *Morse theory*. We illustrate the key idea in Figure 3, and refer to the Appendix C.3 for more detail. We apply our loss in several experiments, leaving a detailed comparison to the discretization approach to a future study.

Morse theory relates the topology of a manifold to the critical points of functions defined on that manifold. In essence, the topology of a sub-level set $\Omega(t) = \{x \in \mathcal{D} | f(x) \leq t\}$ changes only when t passes through a critical value of f. Rooted in Morse theory is the *surface network*, which is a graph with vertices as critical points and edges as *integral paths* (see Figure 3). This and related graphs compactly represent topological information and find many applications in computer vision, graphics, and geometry (Biasotti et al., 2008a; Rana, 2004). However, existing algorithms construct them on discrete representations. First, we extend the construction of a surface network to INSs by leveraging automatic differentiation. This is detailed in Appendix C.2 and illustrated in Figure 11. Second, we construct a differentiable connectedness loss by relaxing the inherently discrete constraint. The key insight is that connected components of Ω are born at minima, destroyed at maxima, and connected via saddle points. Using an augmented edge-weighted graph built from the surface network, we first identify and then connect disconnected components by penalizing the value of f at certain saddle points, detailed in Appendix C.3. Our connectedness loss is summarized in Algorithm 1.

C.2. Surface network

We start by briefly introducing the necessary background from differential topology and Morse theory and refer to Biasotti et al. (2008a;b); Rana (2004) for a more thorough introduction.

Morse theory. Let M be a smooth compact n-dimensional manifold without a boundary, and $f : M \mapsto \mathbb{R}$ a twice continuously differentiable function defined on it. Let $H_f(p)$ denote the Hessian matrix of f at $p \in M$. A critical point $p \in M$ is *non-degenerate* if $H_f(p)$ non-singular. For a non-degenerate critical point p, the number of negative eigenvalues of the Hessian is called the *index* of p. f is called a *Morse function* if all the critical points of f are non-degenerate. f is sometimes called a *simple* Morse function if all the critical points p have different values $f(p)$. (Simple) Morse functions are dense in continuous functions. Under mild assumptions most NNs are Morse functions (Kurochkin, 2021).

Surface networks are a type of graph used in Morse theory to capture topological properties of a sub-level set. They originated in geospatial applications to study elevation maps $f : \mathcal{X} \subset \mathbb{R}^2 \mapsto \mathbb{R}$ on bounded 2D domains. More precisely, a surface network is a graph whose vertices are the critical points of f connected by edges which represent integral paths. An integral path $\gamma : \mathbb{R} \mapsto M$ is everywhere tangent to the gradient vector field: $\partial\gamma/\partial s = \nabla f(\gamma(s))$ for all $s \in \mathbb{R}$. Both ends of an integral path $\lim_{s \mapsto \pm\infty} \gamma(s)$ are at critical points of f. There exist classical algorithms to find surface networks on grids, meshes, or other discrete representations (Rana, 2004; Biasotti et al., 2008b).

We extend the construction of the surface network to an INS represented by a NN f leveraging automatic differentiation in the following steps (illustrated in Figure 11).

1. **Find critical points.** Initialize a large number of points $X \subset \mathcal{X}$, e.g. by random or adaptive sampling. Minimize the norm of their gradients using gradient descent: $\min_X \sum_{x \in X} ||\nabla f(x)||_2^2$. After reaching a stopping criterion, remove points outside of the domain and non-converged candidate points whose gradient norm exceeds some threshold. Cluster the remaining candidate points. We use DBSCAN (Ester et al., 1996).
2. **Characterize critical points** by computing the eigenvalues of their Hessian matrices $H_f(x)$. Minima have only positive eigenvalues, maxima only negative eigenvalues, and saddle points have at least one positive and one negative eigenvalue.
3. **Find integral paths.** From each saddle point, start $2 \dim X$ integral paths, each tangent to a Hessian matrix eigenvector with a positive/negative eigenvalue. Follow the positive/negative gradient until reaching a local maximum/minimum or leaving the domain.
4. **Construct surface network** as a graph $G = (V, E)$, where the set of vertices V consists of the critical points from step 1 and the set of edges E from step 3.

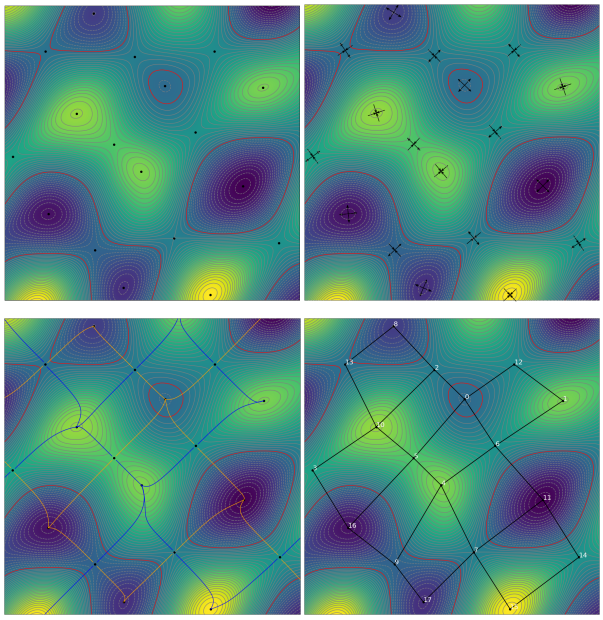


Figure 11: The four steps of constructing the surface network from left to right. (1) Find the critical points by doing gradient descent to the minimum of the gradient norm of the input points. (2) Characterize critical points via analyzing the eigenvalues of the points' Hessians. (3) Connect the saddle points to the adjacent critical points via gradient ascent/descent. (4) Construct the surface network graph with edges corresponding to the ascents/descents from the saddle points.

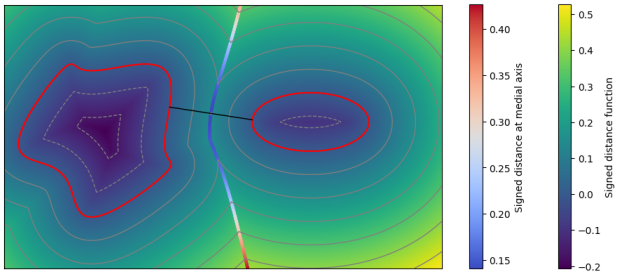


Figure 12: A signed distance function that describes a shape (in red) with two connected components (an ellipse on the right and a wavy pentagon on the left). The contour colors in- and outside the shape increase according to the eikonal equation $||\nabla f(x)|| = 1$ and are described by the gray level sets and the right colorbar. The left colorbar describes the SDF values at the medial axis which is a line of discontinuity, since at each point of the medial axis, the distance to both components is equal. The black line marks the shortest distance between the two connected components. This line crosses the medial axis at the medial axis point with minimum elevation. The point of intersection is exactly half of the distance between the two components.

C.3. Connectedness loss

In Morse theory components of the sub-level set appear at minima, disappear at maxima, and connect through saddle points. Morse theory only assumes that the function is Morse, but on (approximate) SDFs, saddle points can be associated with the medial axis.

Signed distance function (SDF) $f : \mathcal{X} \mapsto \mathbb{R}$ of a shape Ω gives the (signed) distance from the query point x to the closest boundary point:

$$f(x) = \begin{cases} d(x, \partial\Omega) & \text{if } x \in \Omega^c \text{ (if } x \text{ is outside the shape),} \\ -d(x, \partial\Omega) & \text{if } x \in \Omega \text{ (if } x \text{ is inside the shape).} \end{cases} \tag{8}$$

A point $x \in \mathcal{X}$ belongs to the *medial axis* if its closest boundary point is not unique. The gradient of an SDF obeys the eikonal equation $||\nabla f(x)|| = 1$ everywhere except on the medial axis where the gradient is not defined. Figure 12 depicts an SDF for a shape with two connected components. In INS, the SDF is approximated by a NN with parameters θ: $f_\theta \approx f$.

Intuition. Figure 12 shows an exact SDF with two connected components (CCs) (in red) and serves as an entry point to presenting the connectedness loss in more detail. The shortest line (in black) between the two CCs intersects the medial axis at x'. At this intersection, both directions along the shortest line are descent directions and the restriction of f to the medial axis has a local minimum (i.e., has two ascent directions). Nonetheless, this point x' is not a proper saddle point, since the gradient $\nabla f(x')$ is not well-defined. However, we can expect the approximate SDF $f_\theta \in C^2$ to have a saddle at x'. To connect two CCs along the shortest path, we can consider the medial axis, i.e. the saddle points of the approximate SDF. Therefore, we build a connectedness loss by penalizing the value of f at the saddle points in a certain way.

Multiple saddle points between two connected components. In general, there is no reason to expect there is a unique saddle point between two CCs so any or all of the multiple saddle points can be used to connect the CCs. Many approaches are generalized by a penalty weight p_i for each saddle point x_i. E.g., one simple solution is to pick the saddle point on the shortest path between the CCs amounting to a unit penalty vector p. Another solution is to penalize all saddle points between the two CCs equally. We pick the penalty p to be inversely proportional to the distance d between two shape boundaries, i.e. $p \sim \frac{1}{d}$. This implies that the shorter the distance between two CCs via a saddle point, the higher its penalty and the more incentive for the shape to connect there.

Shortest paths using distance weighted edges. We construct the surface network of f_θ as explained in Section C.2. We modify this graph by weighting the edges with the distances between the nodes. We weigh the edges that connect nodes of the same CC with 0. In total, the weighted graph G_w allows us to find the shortest paths between pairs of CCs using graph traversal.

Robustness. Thus far we assumed that (i) f_θ is a close approximation of the true SDF f and (ii) that we find the exact surface network of f_θ. However, in practice, these assumptions rarely hold, so we introduce two modifications to aid the robustness.

Robustness to SDF approximation. The assumption that f_θ is a close approximation of the true SDF is easily violated during the initial stages of training or when the shape undergoes certain topological changes. For a true SDF, the shortest path between two CCs crosses the medial axis only once, so one would expect two CCs to connect via a single saddle point. For an approximate SDF, the shortest path might contain multiple saddle points. However, this simply corresponds to multiple hops in the graph G_w which does not pose additional challenges. We choose to penalize only those saddle points that are adjacent to the shape so that the shape grows outward. Alternatively, one could penalize all the saddle points on the entire shortest path. While this can cause new components to emerge in-between the shapes, this and other options are viable choices that can be investigated further.

Robustness to surface network approximation. So far, we also assume that we extract the exact surface network of f_θ (independent of whether it is an exact or approximate SDF). However, due to numerical limitations, it may not contain all critical points or the correct integral paths. This can cause not being able to identify a path between CCs. In the extreme case, the erroneously constructed surface network might be entirely empty, in which case there is no remedy. To improve the robustness against milder cases, we augment G_w with edges between all pairs of critical points that are outside of the shape. The edge weights are set to the Euclidean distances between the points, resulting in the augmented weighted graph G_a. This improves the likelihood that there always exists at least one path between any two CCs.

Algorithm. Once we have computed the penalty weights, we normalize them for stability and compute the loss. Putting it all together we arrive at Algorithm 1.

Limitations. As mentioned in Section 5 and Appendix A.4, our current approach is computationally costly due to building the surface network and traversing the augmented graph in every epoch. While we manage to update and reuse these structures in some cases, doing this reliably requires further investigation. Furthermore, the requisite robustness of the practical implementation has led to deviations from the theoretical foundations. Overall, there is a compelling motivation for future research to address both theoretical and practical aspects, alongside exploring incremental adjustments or entirely novel methodologies.

Algorithm 1 Connectedness loss

Input: augmented weighted surface network G_a constructed from f_θ
Output: connectedness loss $l_{\text{connectedness}}$

for each node k **do**
  initialize penalty $p_k = 0$
  if k is adjacent to a component c_l **then**
    for each pair of connected components $\{c_i, c_j\}$ **do**
      compute d_{ij} as the length of the shortest path in G_a connecting any node in c_i and any node in c_j via node k
      add to penalty of k according to the distance $p_k = p_k + \frac{1}{d_{ij}+\epsilon}$
    end for
  end if
end for
for each node k **do**
  normalize the penalty $p_k = \frac{p_k}{\sum_l p_l}$
end for
compute the loss $l_{\text{connectedness}} = \sum_k p_k f(x_k)$

D. Geometric constraints and regularization of neural fields

D.1. Geometric constraints

In Table 5, we provide a non-exhaustive list of more constraints relevant to GINNs.

D.2. Regularization of Neural fields

Regularization methods have been proposed to counter the ill-posedness in geometry problems. These include leveraging ground-truth normals (Atzmon & Lipman, 2021) and curvatures (Novello et al., 2022), minimal surface property (Atzmon & Lipman, 2021), and off-surface penalization (Sitzmann et al., 2020). A central effort is to achieve the distance field property of the scalar field for which many regularization terms have been proposed: eikonal loss (Gropp et al., 2020) and divergence loss (Ben-Shabat et al., 2022) among others (Yang et al., 2023; Ma et al., 2023; Marschner et al., 2023).

Inductive bias. In addition to explicit loss terms, the architecture, initialization, and optimizer can also limit or bias

Constraint	Comment
Volume	Non-trivial to compute and differentiate for level-set function (easier for density).
Area	Non-trivial to compute, but easy to differentiate.
Minimal feature size	Non-trivial to compute, relevant to topology optimization and additive manufacturing.
Symmetry	Typical constraint in engineering design, suitable for encoding.
Tangential	Compute from normals, typical constraint in engineering design.
Parallel	Compute from normals, typical constraint in engineering design.
Planarity	Compute from normals, typical constraint in engineering design.
Angles	Compute from normals, relevant to additive manufacturing.
Curvatures	Types of curvatures, curvature variations, and derived energies.
Betti numbers	Topological constraint (number of d-dimensional holes), surface network might help.
Euler characteristic	Topological constraint, surface network might help.

Table 5: A non-exhaustive list of geometric and topological constraints relevant to GINNs but not considered in this work.

the learned shapes. For example, typical INS are limited to watertight surfaces without boundaries or self-intersections (Chibane et al., 2020; Palmer et al., 2022). ReLU networks are limited to piece-wise linear surfaces and typically biased toward low frequencies (Tancik et al., 2020), while fourier-feature encoding (Tancik et al., 2020) and sinusoidal activations can change the bias toward higher frequencies (Sitzmann et al., 2020).

E. Diversity

Concavity. We elaborate on the aforementioned *concavity* of the diversity aggregation measure with respect to the distances. We demonstrate this in a basic experiment in Figure 13, where we consider the feasible set $\mathcal{K}$ as part of an annulus. For illustration purposes, the solution is a point in a 2D vector space $f \in \mathcal{X} \subset \mathbb{R}^2$. Consequentially, the solution set consists of N such points: $S = \{f_i \in \mathcal{X}, i = 1, \dots, N\}$. Using the usual Euclidean distance $d_2(f_i, f_j)$, we optimize the diversity of S within the feasible set $\mathcal{K}$ using minimal aggregation measure

$$\delta_{\text{min}}(S) = \left(\sum_i \left(\min_{j \neq i} d_2(f_i, f_j) \right)^p \right)^{1/p}, \qquad (9)$$

as well as the total aggregation measure

$$\delta_{\text{sum}}(S) = \left(\sum_i \left(\sum_j d_2(f_i, f_j) \right)^p \right)^{1/p}. \qquad (10)$$

Using different exponents $p \in \{1/2, 1, 2\}$ illustrates how δ_{min} covers the domain uniformly for $0 \leq p \leq 1$, while clusters form for $p > 1$. The total aggregation measure

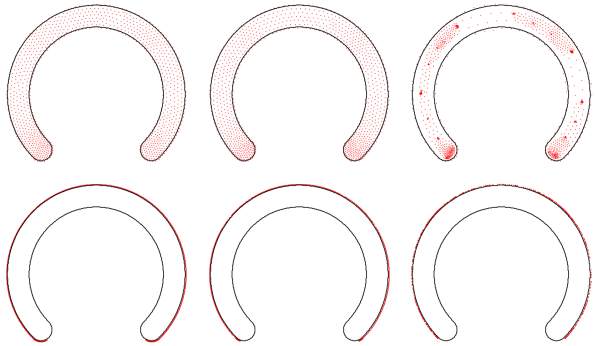


Figure 13: A visual comparison of different diversity losses in a simple 2D example ($\mathcal{F} = \mathbb{R}^2$ and the feasible set $\mathcal{K}$ is the partial annulus). Each point $f \in \mathcal{F}$ represents a candidate solution. The points are optimized to maximize the diversity within the feasible set. The top row shows the *minimal aggregation* δ_{min} as defined in Equation 9. The bottom row shows the *total aggregation* δ_{sum} as defined in Equation 10. Each column uses a different exponent $p \in \{0.5, 1, 2\}$. For $0 \leq p \leq 1$ the minimal aggregation diversity δ_{min} is concave meaning it favors increasing smaller distances over larger distances. This leads to a uniform coverage of the feasible set. In contrast, the δ_{min} is convex for $p \geq 1$ as indicated by the formed clusters for $p = 2$. Meanwhile, δ_{sum} pushes the points to the boundary of the feasible set for all p.

always pushes the samples to the extremes of the domain. Notice that in contrast to these point-samples, the distance measure for shape must be a shape-distance or a measure of dissimilarity. In practice for our shapes, we use a Monte-Carlo estimate of the chamfer discrepancy.

E.1. Overview

Adding an explicit diversity loss not only helps to avoid mode collapse, it also increases the sample diversity even for models that do not suffer from mode-collapse.
Many scientific disciplines require to measure the diversities of sets which has resulted in a range of definitions of diversity (Parreño et al., 2021; Enflo, 2022; Leinster & Cobbold, 2012). Most start from a *distance* $d : \mathcal{F}^2 \mapsto [0, \infty)$, which can be transformed into the related *dissimilarity*. *Diversity* $\delta : 2^{\mathcal{F}} \mapsto [0, \infty)$ is then the collective dissimilarity of a set (Enflo, 2022), aggregated in some way. In the following, we describe these two aspects: the distance d and the aggregation into the diversity δ.

Aggregation. Adopting terminology from Enflo (2022), we use the *minimal aggregation measure*:

$$\delta(S) = \left(\sum_i \left(\min_{j \neq i} d(f_i, f_j) \right)^{1/2} \right)^2 . \tag{11}$$

This choice is motivated by the *concavity* property, which promotes uniform coverage of the available space, as depicted in Figure 13. Section 4.2 demonstrates that adding this to the training objective suffices to counteract mode-collapse. Note, that Equation 11 is well-defined only for finite sets (in practice, a batch) and we leave the consideration of diversity on infinite sets, especially with manifold structure, to future research.

Distance. A simple choice for measuring the distance between two functions is the L^2 function distance $d_2(f_i, f_j) = \sqrt{\int_{\mathcal{X}} (f_i(x) - f_j(x))^2 \, \mathrm{d}x}$. However, recall that we ultimately want to measure the distance between the shapes, not their implicit function representations. For example, consider a disk and remove its central point. While we would not expect their shape distance to be significant, the L^2 distance of their SDFs is. This is because local changes in the geometry can cause global changes in the SDF. For this reason, we modify the distance (derivation in Appendix E) to only consider the integral on the shape boundaries $\partial\Omega_i, \partial\Omega_j$ which partially alleviates the globality issue:

$$d(f_i, f_j) = \sqrt{\int_{\partial\Omega_i} f_j(x)^2 \, \mathrm{d}x + \int_{\partial\Omega_j} f_i(x)^2 \, \mathrm{d}x} \, . \tag{12}$$

If f_j is an SDF then $\int_{\partial\Omega_i} f_j(x)^2 \, \mathrm{d}x = \int_{\partial\Omega_i} \min_{x' \in \partial\Omega_j} ||x - x'||_2^2 \, \mathrm{d}x$ (analogously for f_i) and d is closely related to the *chamfer discrepancy* (Nguyen et al., 2021). We note that d is not a metric distance on functions, but recall that we care about the geometries they represent. Using appropriate boundary samples, one may also directly compute a geometric distance, e.g., any point cloud distance (Nguyen et al., 2021). However, the propagation of the gradients from the geometric boundary to the function requires the consideration of boundary sensitivity (Berzins et al., 2023), which we leave for future work.

Distance. We detail the derivation of our geometric distance. We can partition $\mathcal{X}$ into four parts (one, both or neither of the shape boundaries): $\partial\Omega_i \setminus \partial\Omega_j, \partial\Omega_j \setminus \partial\Omega_i, \partial\Omega_i \cap \partial\Omega_j, \mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)$. Correspondingly, the integral of the L^2 distance can also be split into four terms. Using $f(x) = 0 \; \forall x \in \partial\Omega$ we obtain

$$\begin{aligned}
d_2^2(f_i, f_j) &= \int_{\mathcal{X}} (f_i(x) - f_j(x))^2 \, \mathrm{d}x \\
&= \int_{\partial\Omega_i \setminus \partial\Omega_j} (0 - f_j(x))^2 \, \mathrm{d}x \\
&\quad + \int_{\partial\Omega_j \setminus \partial\Omega_i} (f_i(x) - 0)^2 \, \mathrm{d}x \\
&\quad + \int_{\partial\Omega_i \cap \partial\Omega_j} (0 - 0)^2 \, \mathrm{d}x \\
&\quad + \int_{\mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)} (f_i(x) - f_j(x))^2 \, \mathrm{d}x \\
&= \int_{\partial\Omega_i \setminus \partial\Omega_j} f_j(x)^2 \, \mathrm{d}x \\
&\quad + \int_{\partial\Omega_j \setminus \partial\Omega_i} f_i(x)^2 \, \mathrm{d}x + \\
&\quad \int_{\mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)} (f_i(x) - f_j(x))^2 \, \mathrm{d}x \\
&= \int_{\partial\Omega_i} f_j(x)^2 \, \mathrm{d}x + \int_{\partial\Omega_j} f_i(x)^2 \, \mathrm{d}x + \\
&\quad \int_{\mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)} (f_i(x) - f_j(x))^2 \, \mathrm{d}x
\end{aligned} \tag{13}$$