GEOMETRY-INFORMED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Geometry is a ubiquitous tool in 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) – a framework for training shape-generative neural fields *without data* by leveraging user-specified design requirements in the form of objectives and constraints. By adding *diversity* as an explicit constraint, GINNs avoid mode-collapse and can generate multiple diverse solutions, often required in geometry tasks. Experimentally, we apply GINNs to several validation problems and a realistic 3D engineering design problem, showing control over geometrical and topological properties, such as surface smoothness or the number of holes. These results demonstrate the potential of training shape-generative models without data, paving the way for new generative design approaches without large datasets.

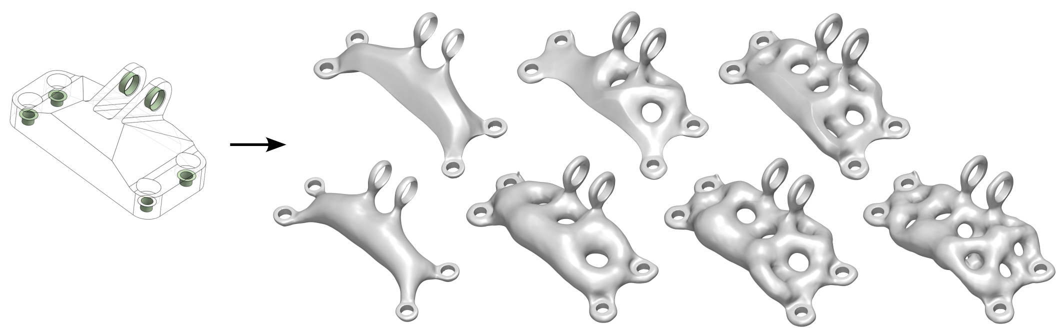

Figure 1: We train geometry-informed neural networks to *produce shapes satisfying geometric design requirements*. For example, we generate parts that connect the cylindrical interfaces within the sketched design region depicted on the left. To highlight the *user's control over the problem* and the solutions, we specify different additional requirements on the number of holes and surface smoothness. By complementing the design requirements with a diversity constraint, we can train a shape-generative model *without data* as illustrated in Figures 3 and 6.

1 INTRODUCTION

Recent advances in deep learning have revolutionized fields with abundant data, such as computer vision and natural language processing. However, the scarcity of large datasets in many other domains, including 3D computer graphics, design, engineering, and physics, restricts the use of advanced supervised learning techniques, necessitating the exploration of alternative learning strategies.

Fortunately, these disciplines are often equipped with formal problem descriptions, such as objectives and constraints. Previous works for PDEs (Raissi et al., 2019), molecular science (Noé et al., 2019), and combinatorial optimization (Bengio et al., 2021) demonstrate these can suffice to train models even in the absence of any data. The success of these data-free approaches motivates an analogous attempt in geometry, raising the question: *Is it possible to train a shape-generative model on objectives and constraints alone, without relying on any data?*

We address this question by introducing *geometry-informed neural networks* or *GINNs*. GINNs are trained to satisfy specified design constraints and to produce feasible shapes without any training samples. A GINN solves a topology optimization problem using *neural fields*, which offer detailed, smooth, and topologically flexible geometry representations, while being compact to store. This setup is analogous to physics-informed neural networks but with a high number of varied constraints: differential, integral, geometrical, and topological.

In contrast to both physics-informed neural networks and classical topology optimization, GINNs allow to generate multiple solutions by enforcing solution *diversity* as an explicit constraint. This is of high interest when applied to problems with solution multiplicity, e.g., induced by under-determinedness or near-optimality common in geometry problems. To connect back to our research question: with GINNs we can train neural fields that satisfy user-specified design constraints, and by adding diversity as an explicit constraint, we can generate a multiplicity of solutions. GINNs can thus be used as shape-generative models trained purely on constraints and without data.

We take several steps to demonstrate GINNs experimentally. We formulate a tractable learning problem using constrained optimization and by converting constraints into differentiable losses. After solving several introductory problems, we proceed to a realistic 3D engineering design problem. Figure 1 illustrates this task of designing a jet-engine lifting bracket, or geometrically – connecting cylindrical interfaces within the given design region. We show different GINNs trained with various additional smoothness and topology requirements. Figure 6 shows a GINN trained on the same task but with an additional diversity constraint. Surprisingly, this induces a structured latent space, with generalization capacity and interpretable directions.

We show that training shape-generative networks using constraints and objectives without data is a feasible learning strategy, paving the way for new generative design approaches without large datasets. Our main contributions are summarized as follows:

1. We introduce GINN - a framework for training shape-generative neural fields without data by leveraging design constraints and avoiding mode-collapse using a diversity constraint.
2. We apply GINNs to several validation problems and a realistic 3D engineering design problem, showing control over geometrical and topological properties. [1]

2 Related work

We begin by reviewing and relating three important facets of GINNs: theory-informed learning, neural fields, and generative modeling.

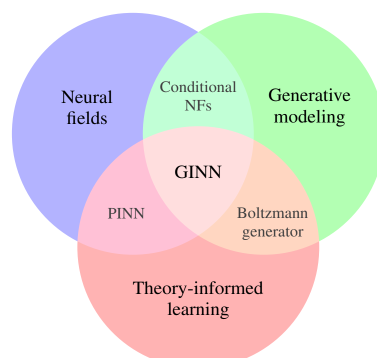

Figure 2: GINNs build on neural fields, generative modeling, and theory-informed learning.

2.1 Theory-informed learning

Theory-informed learning has introduced a paradigm shift in scientific discovery by using 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). Geometric deep learning (Bronstein et al., 2021) introduces a principled way to characterize problems based on symmetry and scale separation principles, e.g. group equivariances or physical conservation laws.

Notably, most works operate in the typical deep learning regime, i.e., with an abundance of data. However, in theory-informed learning, training on data can be replaced by training with objectives and constraints. 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. For example, in Boltzmann generators (Noé et al., 2019), f is a probability function parameterized by a neural network to approximate an intractable target distribution. Another example is combinatorial optimization where

[1] The code will be made public upon acceptance.

$f \in \{0,1\}^N$ is often sampled from a probabilistic neural network (Bello et al., 2016; Bengio et al., 2021; Sanokowski et al., 2024).

Physics-informed neural networks (PINNs) (Raissi et al., 2019) are a prominent example of neural optimization. In PINNs, f is a function that must minimize the violation o of a partial differential equation (PDE), the initial and boundary conditions, and, optionally, some measurement data. Since PINNs can incorporate noisy data and are mesh-free, they hold the potential to overcome the limitations of classical mesh-based solvers for high-dimensional, parametric, and inverse problems. This has motivated the study of the PINN architectures, losses, training, initialization, and sampling schemes (Wang et al., 2023). We further refer to the survey by Karniadakis et al. (2021).
Same as PINNs, GINNs use neural fields to represent the solution. Consequentially, we also observe that some best practices of training PINNs (Wang et al., 2023) transfer to training GINNs. However, PINNs may suffer from ill numerical properties due to minimizing the squared residual of the strong-form different to classical PDE solvers (Rathore et al., 2024; Ryck et al., 2024). In contrast, GINNs share the same underlying formulation and numerical properties as classical topology optimization methods. In addition to a high number of various constraints (differential, integral, geometrical, and topological), geometric problems often require solution multiplicity, motivating the generative extension.

2.2 Neural fields

A *neural field* (NF) (also coordinate-based NN, implicit neural representation (INR)) is a NN (typically a multilayer-perceptron) 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, continuity, and access to automatic differentials. They have seen widespread success in representing and generating a variety of signals, including shapes (Park et al., 2019; Chen & Zhang, 2019; Mescheder et al., 2019), scenes (Mildenhall et al., 2021), images (Karras et al., 2021), audio, video (Sitzmann et al., 2020), and physical quantities (Raissi et al., 2019). For a more comprehensive overview, we refer to the survey by Xie et al. (2022).

Implicit neural shapes (INSs) represent geometries through scalar fields, such as occupancy (Mescheder et al., 2019; Chen & Zhang, 2019) or signed-distance (Park et al., 2019; Atzmon & Lipman, 2020). In addition to the properties of NFs, INSs also enjoy topological flexibility supporting shape reconstruction and generation. We point out the difference between these two training regimes. In the generative setting, the training is supervised on the ground truth scalar field of every shape (Park et al., 2019; Chen & Zhang, 2019; Mescheder et al., 2019). However, in surface reconstruction, i.e., finding a smooth surface from a set of points measured from a single shape, no ground truth is available and the problem is ill-defined (Atzmon & Lipman, 2020; Berger et al., 2016).

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), divergence loss (Ben-Shabat et al., 2022), directional divergence loss (Yang et al., 2023), level-set alignment (Ma et al., 2023), or closest point energy (Marschner et al., 2023). The distance field property can be expressed as a PDE constraint called *eikonal equation* $|\nabla f(x)| = 1$, establishing a relation of regularized INSs to PINNs (Gropp et al., 2020).

Inductive bias. In addition to explicit loss terms, the architecture, initialization, and optimizer can also limit or bias the learned shapes. For example, typical INSs 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 together with gradient descent are biased toward low frequencies (Tancik et al., 2020). Fourier-feature encoding (Tancik et al., 2020), sine activations (Sitzmann et al., 2020), and wavelet activations (Saragadam et al., 2023) allow to control this frequency bias. Similarly, initialization techniques are important to converge toward desirable optima (Sitzmann et al., 2020; Atzmon & Lipman, 2020; Ben-Shabat et al., 2022; Wang et al., 2023).

2.3 (DATA-FREE) GENERATIVE MODELING

Generative modeling (Kingma & Welling, 2013; Goodfellow et al., 2014; Rezende & Mohamed, 2015; Tomczak, 2021) is almost exclusively performed in a data-driven (i.e., supervised) setting to capture and sampling from the underlying data-distribution. However, notable exceptions exist.

Boltzmann generators (Noé et al., 2019) are a prominent example of *data-free* generative models. They are trained to capture the Boltzmann distribution associated with an energy landscape. In the generative setting, GINNs also learn a distribution minimizing an energy as an implicit combination of constraint violations and objectives. However, Boltzmann generators avoid mode-collapse using an entropy-regularizing term which presupposes invertibility making them not directly applicable to function spaces. Instead, GINNs use a more general diversity term to hinder mode-collapse over the function space of shapes.

Conditional neural fields allow for generative modeling of functions. By conditioning a base network F on a modulation (i.e, latent) variable z, a conditional NF encodes multiple fields simultaneously: $f(x) = F(x; z)$. The different choices of the conditioning mechanism lead to a zoo of architectures, including input concatenation (Park et al., 2019), hypernetworks (Ha et al., 2017), modulation (Mehta et al., 2021), and attention (Rebain et al., 2022). These can be classified into global and local mechanisms, which also establishes a connection between conditional NFs and operator learning (Wang et al., 2024). For more detail we refer to Xie et al. (2022); Rebain et al. (2022); Wang et al. (2024).

Generative design refers to computational methods that automatically conduct design exploration under constraints set by designers (Jang et al., 2022). It holds the potential of streamlining innovative solutions, e.g., in material design, architecture, or engineering. In particular, GINNs can be seen as solving the general task of *topology optimization* – finding the material distribution that minimizes a specified objective subject to constraints. However, while classical methods optimize a single shape directly, we optimize a GINN that generates diverse feasible shapes. This encourages design space exploration and supports downstream tasks, while allowing to incorporate even sparse data samples, if available. While generative design datasets are not abundant, deep learning has previously shown promise in material design and topology optimization. For more detail, we refer to surveys on generative models in engineering design (Regenwetter et al., 2022) and topology optimization via machine learning (Shin et al., 2023).

3 METHOD

Consider the metric space $(d, \mathcal{F})$ of functions, such as those representing a shape or a PDE solution. Let the set of constraints define the feasible set $\mathcal{K} = \{f \in \mathcal{F} | c_i(f) = 0, i = 1..m\}$. Additionally, let the geometric problem be equipped with an objective $o : \mathcal{F} \mapsto \mathbb{R}$. Selecting the objectives and constraints of a geometric nature lays the foundation for a *GINN*, which is trained to produce an optimal feasible solution by solving $\min_{f \in \mathcal{K}} o(f)$. A key feature of geometric problems is that one is often interested in finding different near-optimal solutions, for example, due to incompleteness, uncertainty, or under-determinedness in the problem specification (e.g. see Figure A.4). This motivates making GINN *generate* a set of sufficiently diverse near-optimal solutions $S \subset \mathcal{K}$:

$$\min_{\substack{S \subset \mathcal{K} \\ \delta(S) \geq \delta_{\min}}} O(S) \, . \tag{1}$$

$O(S)$ aggregates the objectives $o(f)$ of all solutions $f \in S$ and δ captures some intuitive notion of *diversity*. It is yet another constraint, however it acts on the entire solution set instead of each solution separately. Section 3.1 first discusses representing shapes as functions, in particular, neural fields, and formulating differentiable constraints. In Section 3.2, we generalize to representing and finding diverse solutions using conditional neural fields.

3.1 FINDING A SOLUTION

Representation of a solution. Let $f : \mathcal{X} \mapsto \mathbb{R}$ be a continuous scalar function on the domain $\mathcal{X} \subset \mathbb{R}^n$. The sign of f *implicitly* defines the shape $\Omega = \{x \in \mathcal{X} | f(x) \leq 0\}$ and its boundary $\partial\Omega = \{x \in \mathcal{X} | f(x) = 0\}$. We use a NN $f = f_\theta$ with parameters θ to represent the implicit

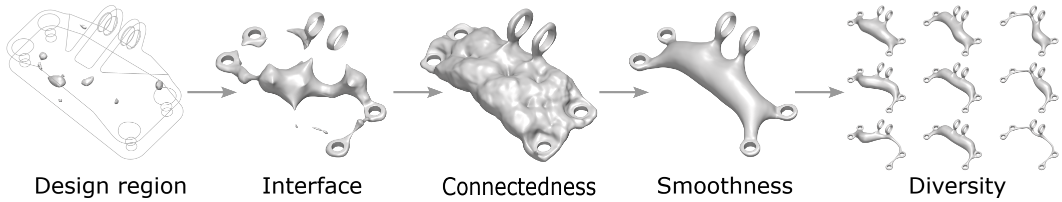

Figure 3: The user can define geometric problems and solve them using the GINN framework. Here, we illustrate the results of progressively adding more design requirements, overall resulting in a shape generative model trained without data.

	Set constraint $c_i(\Omega)$	Function constraint	Constraint violation $c_i(f)$
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	$n(x) = \bar{n}(x) \, \forall x \in \mathcal{I}$	$\frac{\nabla f(x)}{\lvert\lvert \nabla f(x) \rvert\rvert} = \bar{n}(x) \, \forall x \in \mathcal{I}$	$\int_{\mathcal{I}} \left[\frac{\nabla f(x)}{\lvert\lvert \nabla f(x) \rvert\rvert} - \bar{n}(x) \right]^2 \mathrm{d}x$
Topology	Using persistent homology; see Section 4.1 and Appendix E		

Table 1: Geometric constraints used in our main experiment. The shape Ω and its boundary $\partial\Omega$ are represented implicitly by the (sub-)level set of the function f. 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 prescribed *normal* $\bar{n}(x)$. Other interesting constraints are listed in Table 5.

function, i.e. an *implicit neural shape*, due to its memory efficiency, continuity, and differentiability. Nonetheless, the GINN paradigm extends to other representations, as demonstrated in Section 4.2. We additionally require f to approximate the *signed-distance function* (SDF) of Ω (defined in Eq. 21). This alleviates the ambiguity of many implicit functions representing the same geometry and aids the computation of persistent homology, surface point samples, and diversity. In training, the eikonal constraint is treated analogously to the geometric constraints.

Constraints on a solution. To perform gradient-based optimization, we must first ensure each constraint can be written as a differentiable constraint violation $c_i : \mathcal{F} \mapsto \mathbb{R}$. A geometric constraint has the general form $c_i(\Omega, \partial\Omega) = 0$. By representing the shape and its boundary as the (sub-)level-set of the function f, the constraints on the sets can be translated into constraints on f. This in turn allows to formulate differentiable constraint violations c_i, although this choice is not unique. Table 1 shows several examples using the constraints from our main experiment. Some losses are straightforward and some have been previously demonstrated as regularization terms for INSs (see Section 2.2). Section 4.1 discusses two complex losses in more detail: connectedness and smoothness.

3.2 Generating diverse solutions

Representation of the solution set. The *generator* $G : z \mapsto f$ maps a latent variable $z \in Z$ to a function 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: $f(x) = F(x; z)$.

Constraints on the solution set. By adopting a probabilistic view, we extend each constraint violation and the objective to their expected values: $C_i(S) = \mathbb{E}_{z \sim p_Z} [c_i(G(z)]$ and $O(S) = \mathbb{E}_{z \sim p_Z} [o(G(z)]$.

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 constraint*.

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 with 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*:

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

This choice is motivated by the *concavity* property, which promotes uniform coverage of the available space, as depicted in Figure 12. Figure 5(c) illustrates how it counteracts mode-collapse in a geometric problem. However, Equation 2 is well-defined only for finite sets, so, in practice, we apply δ to a batch of k i.i.d. sampled shapes $S_k = \{G(z_i) | z_1, ..., z_k \overset{iid}{\sim} p_Z\}$. 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 F) to only consider the integral on the shape boundaries $\partial\Omega_i, \partial\Omega_j$ which partially alleviates the globality issue:

$$d(f_i, f_j) = \sqrt{\int_{\partial\Omega_i} f_j(x)^2 \, \mathrm{d}x + \int_{\partial\Omega_j} f_i(x)^2 \, \mathrm{d}x} \, . \tag{3}$$

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.

In summary, training a GINN corresponds to solving a constrained optimization problem, i.e. improving the expected objective $O(S)$ and feasibility $C_i(S)$ w.r.t. to each geometric constraint $i = 1..m$ and the diversity constraint $C_{m+1}(S_k) = \max(\delta(S_k) - \delta_{\text{min}}, 0)$. In practice, we convert this into a sequence of unconstrained optimization problems using the augmented Lagrangian method introduced in Section 4.1.

4 Experiments

We demonstrate the proposed GINN framework experimentally on a set of validation problems, proceed with a 2D engineering problem in Appendix A.5 and conclude with a realistic 3D engineering design use case. To the best of our knowledge, data-free shape-generative modeling is an unexplored field with no established baselines, problems, and metrics. In addition to the problems defined and solved in Sections 4.2 and 4.3, we define metrics for each constraint as detailed in Appendix C.1. We use these to perform quantitative ablation studies in Appendix C.2, reserving the primary text for the main findings. We proceed with an overview of key experimental considerations with more implementation and experiment details available in Appendix A.

4.1 EXPERIMENTAL DETAILS

Constrained optimization. To solve the aforementioned constrained optimization problems in Eq. 1, we employ the *augmented Lagrangian method* (ALM). It is well studied in the classical and more recently deep learning literature and balances the feasibility and optimality of the solution by controlling the influence of each constraint while avoiding the ill-conditioning and convergence issues of simpler methods. Specifically, we use an *adaptive* ALM proposed by Basir & Senocak (2023) which uses adaptive penalty parameters μ_i for each constraint to solve problem 1 as the unconstrained optimization problem $\max_\lambda \min_\theta \mathcal{L}(\theta, \lambda, \mu)$ where

$$\mathcal{L}(\theta, \lambda, \mu) := O(S_k(\theta)) + \sum_{i=1}^{m+1} \lambda_i C_i(S_k(\theta)) + \frac{1}{2} \sum_{i=1}^{m+1} \mu_i C_i^2(S_k(\theta)) \, . \tag{4}$$

The multipliers λ_i and the penalty parameters μ_i are updated during training according to Equations (18) - (20). Adaptive ALM allows GINNs to handle different constraints without manual hyperparameter tuning for each loss. Appendix D provides a more detailed introduction and motivation for this approach.

Topology describes properties of a shape that are invariant under deformations, such as the number of connected components or holes. Certain materials and objects display specific topological properties (Moore, 2010; Caplan et al., 2018; Bendsoe & Sigmund, 2011), e.g., *connectedness*, which is a basic requirement for the propagation of forces and by extension manufacturability and structural function. Despite topological properties being discrete-valued, *persistent homology* (PH) is a tool that allows to formulate a differentiable loss. In brief, it identifies topological features (e.g., connected components) and quantifies their *persistence* w.r.t. some *filtration* function. For our implicit shapes, this is the implicit function f itself. Consequentially, the *birth* and *death* of each feature can be matched to a pair of critical points of f. Their values can then be adjusted to achieve the desired topology.
In practice, we follow the standard procedure of first discretizing the continuous function onto a cubical complex. We additionally filter cells outside the given design space to prevent undesirable connections. Throughout this section, we use a constraint that encourages connected shapes with a minimal number of holes. We detail PH and our approach in Appendix E.

Smoothness constitutes another computationally non-trivial design requirement that we consider. Many alternative smoothing energies exist, each leading to different surface qualities (Westgaard & Nowacki, 2001; Song, 2021), but a broad class can be written as the surface integral $\int_{\partial\Omega\setminus\mathcal{I}} e(\kappa_1(x), \kappa_2(x)) \,\mathrm{d}x$ of some curvature expression $e : \mathbb{R}^2 \mapsto \mathbb{R}$. The principal curvatures κ_1 and κ_2, same as other differential-geometric quantities, can be computed from $\nabla_x f$ and $H_x f$ in closed-form (Goldman, 2005). To solve Plateau's problem, we use the *mean curvature* $\kappa_H := \frac{\kappa_1+\kappa_2}{2}$. In the main experiment, we primarily focus on the *surface-strain* $E := \kappa_1^2 + \kappa_2^2$ and a variant thereof $E_{\text{log}} := \log(1 + E)$ to produce visually appealing shapes.

Surface sampling is required to estimate the surface integrals for smoothness and diversity. We first sample points in the envelope and project them onto the surface using Newton iterations. We then repel the points on the surface to achieve a more uniform distribution similar to Yifan et al. (2020). Finally, we exclude points sampled within a small distance to the interface $\mathcal{I}$, as the surface should not change here. We also begin to sample surface points and compute the surface integrals only after a warm-up phase of 500 iterations. In combination, these aspects lead to a lower variance and better convergence.

Models. Across the experiments, we consider several neural field models. We require these to have a well-defined and non-vanishing second derivative w.r.t. the inputs x to compute the surface normals and curvatures. As the simplest models, we use MLPs with a softplus (i.e., differentiable ReLU) activation. These suffice for the simpler validation experiments, but their simplicity bias is limiting for the physics and the 3D experiment. Following the recommendation of Wang et al. (2023), we find SIREN suitable for the physics task. For the main task, we employ WIRE (Saragadam et al., 2023) – a generalization of SIREN that replaces sine with Gabor wavelet activation functions, which are localized in both the spatial and frequency domains. We show a qualitative comparison in Figure 8. As the neural field conditioning mechanism, we always use input concatenation (see Section 2.3), denoting the latent space dimension as $\dim(z)$.

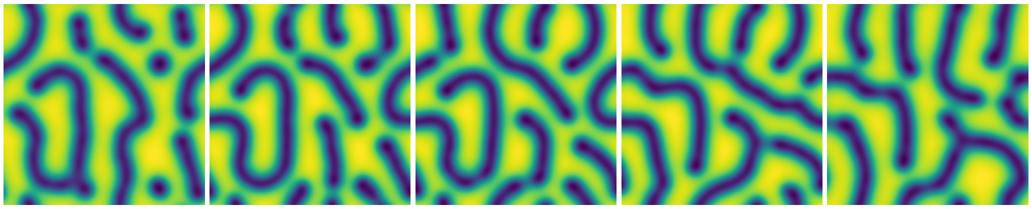

Figure 4: A generative PINN producing Turing patterns that morph during latent space interpolation. This is a result of searching for diverse solutions to an under-determined Gray-Scott system.

4.2 Validation problems

Generative PINN solving an under-determined reaction-diffusion problem. As a primer to geometry tasks, we begin by illustrating solution multiplicity on an under-determined physics system. While most familiar problems in physics are well-defined, cases exist where, e.g., the initial conditions are irrelevant and general PDE solutions are sought, such as in chaotic systems or animations. We first demonstrate how a PINN can also be extended to provide diverse stationary solutions using a system of *reaction-diffusion* with *no initial condition*. Figure 4 illustrates the resulting Turing patterns that continuously morph during latent space traversal. For more detail, we refer to Appendix B.

Plateau's problem to demonstrate GINNs on a well-posed problem. Plateau's problem is to find the surface M 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). Altogether, we represent the surface as $M = \partial\Omega \cap \mathcal{X}$ and the two constraints are: $\Gamma \subset M$ and $\kappa_H(x) = 0 \; \forall x \in M$. The result in Figure 5(a) qualitatively agrees with the known solution.

Parabolic mirror to demonstrate a different geometry representation. Although we mainly focus on the implicit representation, the GINN framework extends to other representations, such as explicit, parametric, or discrete shapes. Here, 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 5(b) 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 combining GINNs and the recent developments in neural rendering techniques.

Obstacle to introduce diversity and connectedness. Consider a 2D rectangular design region $\mathcal{E}$ with a circular obstacle in the middle. The interface $\mathcal{I}$ consists of two vertical line segments and has prescribed outward-facing normals $\bar{n}$. We seek shapes that connect these two interfaces while avoiding the obstacle. Despite this problem admitting infinitely many solutions, the naive application of the generative softplus-MLP leads to mode-collapse. This is mitigated by employing the additional diversity constraint as illustrated in Figure 5(c).

4.3 Engineering design case study

The problem specification is based on an engineering design competition hosted by General Electric and GrabCAD (Kiis et al., 2013). The challenge was to design the lightest possible lifting bracket of a jet engine subject to both physical and geometrical constraints. Here, we focus only on the geometric requirements: the shape must fit in a provided design region $\mathcal{E}$ and attach to five cylindrical interfaces $\mathcal{I}$: a horizontal loading pin and four vertical fixing bolts. Instead of minimizing the volume subject to a linear elasticity PDE constraint, we minimize the surface smoothness E subject to a topological connectedness constraint. Conceptually, this formulation is similar but avoids the need for a PDE solver in the training loop. These requirements are detailed in Table 1 and illustrated in Figs. 1 and 3.

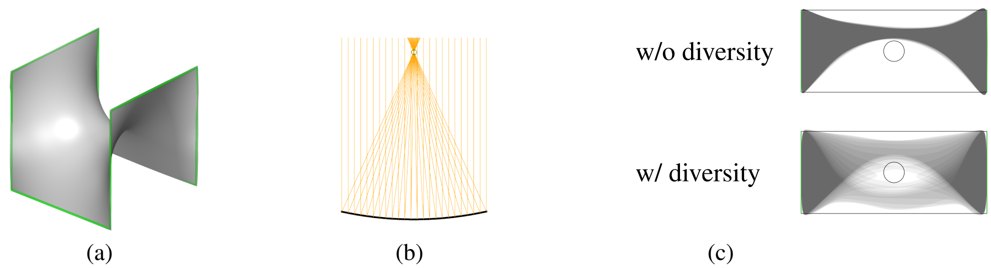

Figure 5: GINN solving three validation problems. (a) Plateau's problem: the unique minimal surface that attaches to the prescribed boundary. (b) Parabolic mirror: the unique surface that collects reflected rays into a single point. (c) Obstacle: connecting the two interfaces within the allowed design region. A superposition of 16 solutions is shown trained with and without a diversity loss, which is required to avoid mode-collapse.

Single solution to the above problem is included in Figure 3. The trained GINN model represents a smooth, singly connected shape attaching to the interfaces while remaining within the given design space.

Ablations. In addition to the qualitative presentation, we perform quantitative evaluation and ablations in Appendix C. Table 3 quantifies the impact of the connectedness constraint and the smoothness objective, as well as some experimental decisions, including the WIRE hyperparameters and eikonal regularization.

User control. We further solve variations of the above problem to highlight the user's ability to tune the problem and the resulting solutions. In particular, we include an additional topology constraint on the number of holes and employ another surface smoothness $E_{\log}$. The variety of produced shapes is illustrated in Figure 1. This also illustrates the robustness of ALM and GINNs to problem variations.

Multiple solutions. Upon the addition of the diversity constraint, GINNs do not only produce multiple solutions, but we also observe the emergence of a *latent space structure*. This is illustrated in Figure 6 using a $\dim(z) = 2$ latent space from which $k = 9$ random samples are drawn every training iteration. Traversing the latent space of the trained GINN produces continuously morphing feasible shapes, i.e. the model *generalizes*. Furthermore, the latent space is *organized*. However, we find that the learned structure depends on the exact setup of the diversity constraint. In particular, we observe a more pronounced organization emerge for larger $\delta_{\min}$ while over-specifying it impacts convergence.

Optimization. The convergence behavior of the aforementioned runs with and without diversity is shown in Figures 10 and 11. Despite having up to seven loss terms, adaptive ALM automatically balances these and minimizes each constraint violation. However, the variance in several losses remains high. This is largely due to the diversity and smoothness terms, which are hard to optimize and increase the necessary number of iterations by roughly a factor of two and five, respectively. In addition, the necessary surface point sampling accounts for roughly 50% of the runtime, motivating improved strategies. Overall, training a single shape takes roughly 10K iterations and 1 hour. Similarly, the discussed diverse model takes 50K iterations and 72 hours.

Surface smoothness. As illustrated in Figure 3, the smoothness term significantly improves the visual quality of the shapes. However, it is a difficult objective. First, the values of the surface-strain E span several orders of magnitude and small deviations in the surface sampling cause large variance. Hence, the strategies described in Section 4.1 are critical to stabilize training. Employing $E_{\log}$ improves convergence further, but also leads to a different surface quality due to the concavity property of $\log$ (cf. top and bottom rows in Figure 1). In previous works optimizing integrals over fixed domains, adaptive ALM deals with the range issue by adjusting location-specific penalties. However, this strategy is not directly applicable to our moving surfaces, which presents a unique future challenge. In addition, curvature is a second-order differential operator and is expected to be ill-conditioned, motivating the use of second-order optimizers to further refine the results (Ryck et al., 2024; Rathore et al., 2024).

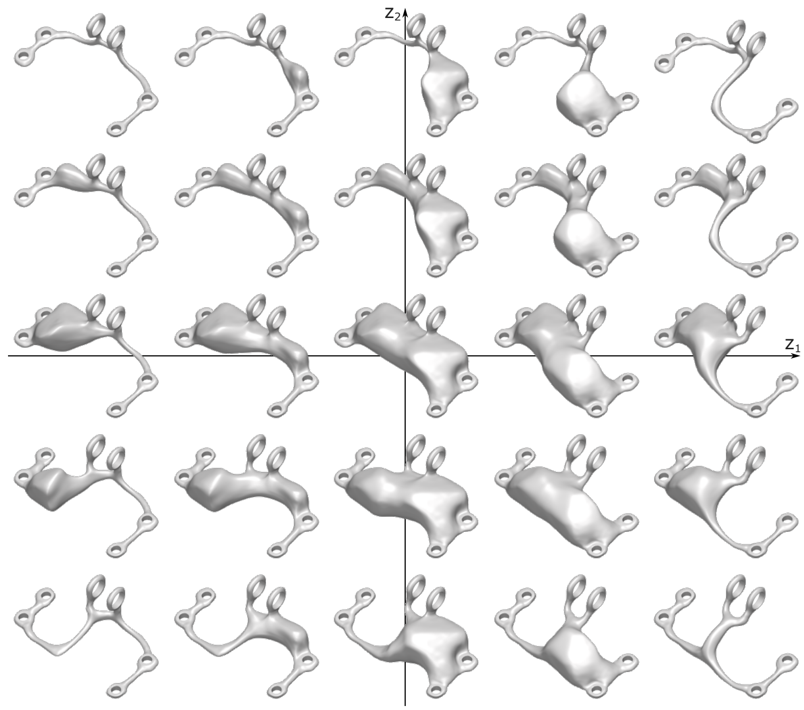

Figure 6: With diversity, GINNs do not only produce multiple solutions but also learn a *latent space structure*. Traversing the 2D latent space continuously morphs solutions, i.e., the model *generalizes*. The latent space is also *organized* – a central bulky shape becomes thinner in the radial direction and the axes can be identified by how the shape connects on the sides. Figure 9 shows a 9×9 version.

5 Conclusion

We have introduced geometry-informed neural networks demonstrating shape-generative modeling driven solely by geometric constraints and objectives. After formulating the learning problem and discussing key theoretical and practical aspects, we applied GINNs to several validation problems and a realistic engineering task.

Limitations and future work. GINNs combine several known and novel components, each of which warrants an in-depth study of theoretical and practical aspects, including alternative shape distances and their aggregation into a diversity, conditioning mechanisms, and a broader range of constraints.

In this work, we focused on building the conceptual framework of GINNs and validating it experimentally. This included a realistic generative design task. However, we considered a modified version of the original task and did not compare to established topology-optimization methods as this required an integration of a PDE solver – a task that future work should address.

Even though ALM is a significant improvement over the naive approach of manually weighted loss terms, the recent literature on multi-objective and second-order optimizers suggest further possible improvements.

Finally, we investigated GINNs in the limit of no data. However, GINNs can integrate partial observations of a single or multiple shapes. This combination of classical and machine learning methods suggests a new approach to generative design in data-sparse settings, which are of high relevance in practical engineering settings.

Ethics statement. Our work aims to advance the field of machine learning and may contribute to its broader societal impact. In addition, there is an ongoing discussion on the rights to data, since data is fundamental to training most current machine learning models. The demonstrated data-free approach to generative modeling brings forth a less explored perspective on this matter. It circumvents the copyright problem and facilitates practitioners who lack exclusive access to datasets. However, for the foreseeable future, the applicability and hence the impact is limited to scientific and engineering applications. The demonstrated results in particular might foster a path toward improved approaches to engineering design.

Reproducibility statement. We provide code to reproduce the main results in the supplementary material. Additionally, we report hyperparameters, and important implementation details to facilitate the reproduction of our results. Upon publication, the code will be made publicly available on GitHub.

REFERENCES

Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

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

Shamsulhaq Basir and Inanc Senocak. An adaptive augmented lagrangian method for training physics and equality constrained artificial neural networks. *arXiv preprint arXiv:2306.04904*, 2023.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. 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.

Martin Philip Bendsoe and Ole Sigmund. *Topology optimization*. Springer, Berlin, Germany, 2 edition, July 2011.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Gael Guennebaud, Joshua Levine, Andrei Sharf, and Claudio Silva. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, pp. 27, 2016.

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

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

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

M. E. Caplan, A. S. Schneider, and C. J. Horowitz. Elasticity of nuclear pasta. *Phys. Rev. Lett.*, 121: 132701, Sep 2018. doi: 10.1103/PhysRevLett.121.132701. URL `https://link.aps.org/doi/10.1103/PhysRevLett.121.132701`.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 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.

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

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

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

J. R. Clough, N. Byrne, I. Oksuz, V. A. Zimmer, J. A. Schnabel, and A. P. King. 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.

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

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

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

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

Ferdinando Fioretto, Pascal Van Hentenryck, Terrence WK Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part V*, pp. 118–135. Springer, 2021.

Rickard Brüel Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In Silvia Chiappa and Roberto Calandra (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.

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

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of Machine Learning and Systems 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3569–3579. PMLR, 13–18 Jul 2020.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, 2017.

Amin Heyrani Nobari, Wei Chen, and Faez Ahmed. 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.

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

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

Shizuo Kaji, Takeki Sudo, and Kazushi Ahara. Cubical ripser: Software for computing persistent homology of image and volume data. *arXiv preprint arXiv:2005.12692*, 2020.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021.

Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

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

Kaspar Kiis, Jared Wolfe, Gregg Wilson, David Abbott, and William Carter. Ge jet engine bracket challenge. `https://grabcad.com/challenges/ge-jet-engine-bracket-challenge`, 2013. Accessed: 2024-05-22.

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

James Kotary and Ferdinando Fioretto. Learning constrained optimization with deep augmented lagrangian methods. *arXiv preprint arXiv:2403.03454*, 2024.

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

B. Ma, J. Zhou, Y. Liu, and Z. Han. 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.

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 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.

Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

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

I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker. 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.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 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.

Mariem Mezghanni, Malika Boulkenafed, André Lieutier, and Maks Ovsjanikov. 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.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Joel E. Moore. The birth of topological insulators. *Nature*, 464(7286):194–198, March 2010. ISSN 1476-4687. doi: 10.1038/nature08916.

Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pp. 36–45. IEEE, 2018.

Kalyan Varma Nadimpalli, Amit Chattopadhyay, and Bastian Alexander Rieck. 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.

T. Nguyen, Q. Pham, T. Le, T. Pham, N. Ho, and B. Hua. 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.

Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinícius da Silva, Hélio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

David Palmer, Dmitriy Smirnov, Stephanie Wang, Albert Chern, and Justin Solomon. DeepCurrents: Learning implicit representations of shapes with boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 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.

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

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

Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. 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.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

Daniel Rebain, Mark J. Matthews, Kwang Moo Yi, Gopal Sharma, Dmitry Lagun, and Andrea Tagliasacchi. Attention beats concatenation for conditioning neural fields. *Trans. Mach. Learn. Res.*, 2023, 2022.

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

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

Tim De Ryck, Florent Bonnet, Siddhartha Mishra, and Emmanuel de Bezenac. An operator preconditioning perspective on training in physics-informed machine learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Sara Sangalli, Ertunc Erdil, Andeas Hötker, Olivio Donati, and Ender Konukoglu. Constrained optimization to train neural networks on critical and under-represented classes. *Advances in neural information processing systems*, 34:25400–25411, 2021.

Sebastian Sanokowski, Wilhelm Berghammer, Sepp Hochreiter, and Sebastian Lehner. Variational annealing on graphs for combinatorial optimization. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Seungyeon Shin, Dongju Shin, and Namwoo Kang. 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.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

Hwijae Son, Sung Woong Cho, and Hyung Ju Hwang. Enhanced physics-informed neural networks with augmented lagrangian relaxation method (al-pinns). *Neurocomputing*, 548:126424, 2023.

Anna Song. Generation of tubular and membranous shape textures with curvature functionals. *Journal of Mathematical Imaging and Vision*, 64(1):17–40, aug 2021. ISSN 1573-7683. doi: 10.1007/s10851-021-01049-9.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

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

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

Thomas Unterthiner, Bernhard Nessler, Calvin Seward, Günter Klambauer, Martin Heusel, Hubert Ramsauer, and Sepp Hochreiter. 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.

Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, 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.

Fan Wang, Huidong Liu, Dimitris Samaras, and Chao Chen. TopoGAN: A topology-aware generative adversarial network. In *Proceedings of European Conference on Computer Vision*, 2020.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks, 2023.

Sifan Wang, Jacob H Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. Bridging Operator Learning and Conditioned Neural Fields: A Unifying Perspective. *arXiv preprint arXiv:2405.13998*, 2024.

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

Geir Westgaard and Horst Nowacki. 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.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. ISSN 1467-8659.

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

Wang Yifan, Shihao Wu, Cengiz Oztireli, and Olga Sorkine-Hornung. Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *CVPR*, 2020.

A IMPLEMENTATION AND EXPERIMENTAL DETAILS

We report additional details on the experiments and their implementation. We run all experiments on a single GPU (one of NVIDIA RTX2080Ti, RTX3090, A40, or P40). For single-shape training, the maximum GPU memory requirements are ca. 9GB for the jet engine bracket and less than a GB for the rest. For multi-shape training the maximum GPU memory requirements are ca. 45GB for the jet engine bracket (9 shapes) and ca. 7GB for the obstacle problem (16 shapes).

A.1 REACTION-DIFFUSION

See Appendix B.

A.2 PLATEAU'S PROBLEM

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

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

The obstacle experiment serves as a proof of concept for including several losses, in particular the connectedness loss.

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 $\bar{n}(\pm 0.9, -0.4 \leq x_2 \leq 0.4) = (\pm 1, 0)$.

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". We train a softplus-MLP with Adam (default settings) and the hyperparameters in Table 2.

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 5(c).

Computational cost. The total training wall-clock time is around 10 minutes for a single shape and approximately 60 minutes for 16 shapes. These numbers are without applying a smoothness loss.

A.5 WHEEL

As an additional engineering use case, we optimize wheels. When we apply the diversity loss with a 2d latent space, we obtain diverse shapes and a structured latent space. The results are depicted in Figure 7.

For this problem, the domain is defined as $X = [-1, 1]^2$. The design space is constrained to the ring-shaped region with an inner radius $r_i = 0.2$ and an outer radius $r_o = 0.8$. The design must adhere to interface constraints involving the inner and outer circles. Additionally, the design must

Hyperparameter	Obstacle (2D)	JEB (3D)
Architecture	Residual-MLP	WIRE
Hidden layers	$[512, 512, 512, 512]$	$[128, 128, 128]$
Activation	softplus	Gabor wavelet
ω_0 for WIRE	n/a	18.0
s_0 for WIRE	n/a	6.0
Learning rate	0.001	0.001
Learning rate schedule	$0.5^{t/1000}$	$0.5^{t/5000}$
Iterations	3000	10000

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 (Obstacle) or 5000 (jet engine bracket) iterations. The hidden layers do not include an input layer of input-dimension $\dim(x) + \dim(z)$, whereas x is a coordinate and z is a modulation vector, and an output layer of output-dimension 1. Interestingly, the WIRE network overall had fewer parameters, while fitting a more complex shape.

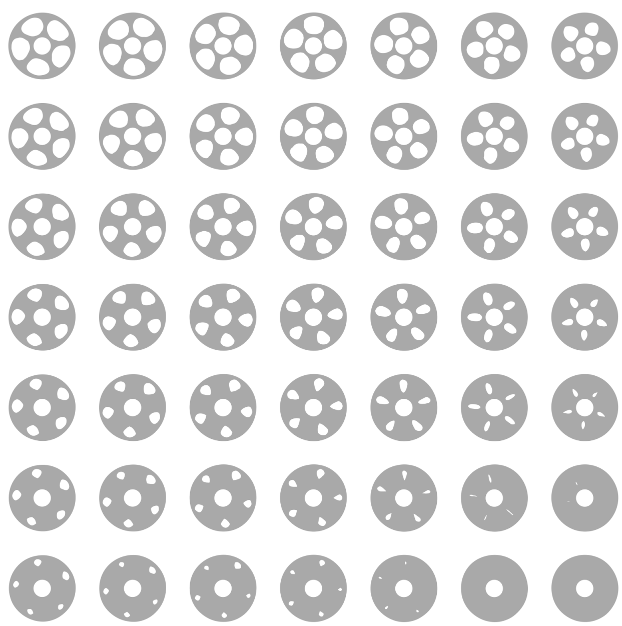

Figure 7: GINNs produce diverse shapes with a structured latent space. The shapes morph continuously into one another when traversing the 2 dimensional latent space. From the top-left to bottom-right, the holes become smaller, while from bottom-left to top-right the holes move inwards.

satisfy a connectedness constraint and a diversity constraint. Furthermore, a 5-fold cyclic symmetry constraint is imposed. This can be implemented as a soft constraint by sampling a point, rotating it by $\frac{2}{5}\pi$ four times, evaluating the implicit function at these five points, and requiring the variance of these values to be zero. Alternatively, a hard constraint using a periodic encoding could be employed to achieve exact symmetry.

A.6 JET ENGINE BRACKET

The jet engine bracket (JEB) is our most complex experiment. We tested different architectures (c.f. Figure 8) and found that WIRE (Saragadam et al., 2023) produced the best results, while being easier to train with the augmented Lagrangian method than softplus-MLP or SIREN (Sitzmann et al., 2020).

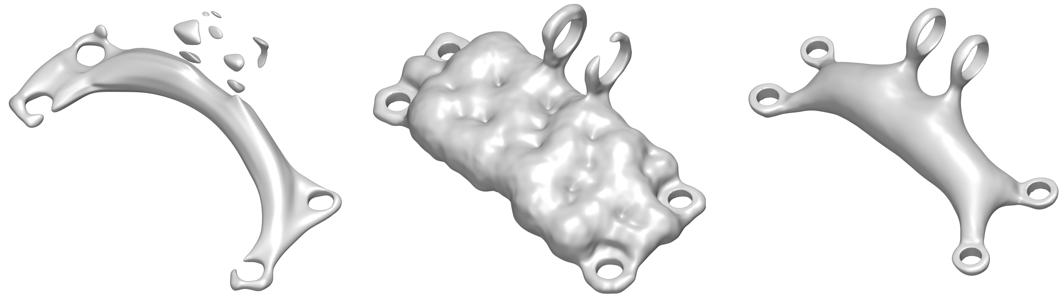

Figure 8: Comparison of architectures trained for 10k epochs to produce a single shape. From left to right: softplus-MLP, SIREN, WIRE. The softplus-MLP is unable to fit the interfaces due to the low-frequency bias. SIREN converges much slower than WIRE, especially at the interfaces, and does not produce a smooth shape.

We train with Adam (default settings) and the hyperparameters summarized in Table 2. To decrease the training time, we use multi-processing to asynchronously create diagnostic plots or computing the PH loss for multiple shapes.

WIRE. For the jet engine bracket settings, early experiments indicated that the softplus-MLP cannot satisfy the given constraints. We therefore employ a WIRE network (Saragadam et al., 2023), which is biased towards higher frequencies of the input signal. As mentioned by the authors, the spectral properties of a WIRE model are relatively robust. Several values for ω_0 and s_0, which control the frequency and scale of the gaussian of the first layer at initialization, were tested. As there was no big difference in the results, we fixed them to $\omega_0 = 18$ and $s_0 = 6$ For more detailed results, we refer to Section C.

Conditioning the model. In the generative GINN setting, we condition WIRE 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 $N = 9$ different latent codes spaced in the interval $Z = [0, 0.1]$ and are resampled every iteration. The results are shown in Figure 9.

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. To facilitate learning we additionally increase the resolution around the interfaces.

B Generative PINNs

Having developed a generative GINN that is capable of producing diverse solutions to an under-determined 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. Such systems were introduced by Turing (1952) to explain how patterns in nature, such as stripes and spots, can form as a result of a simple physical process of reaction and diffusion of two substances. A celebrated 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:

$$\frac{\partial u}{\partial t} = D_u \Delta u - uv^2 + \alpha(1-u) \,, \quad \frac{\partial v}{\partial t} = D_v \Delta v + uv^2 - (\alpha+\beta)v \,. \tag{5}$$

This PDE describes the concentration u, v of two substances U, V undergoing the chemical reaction $U + 2V \rightarrow 3V$. The rate of this reaction is described by uv^2, while the rate of adding U and

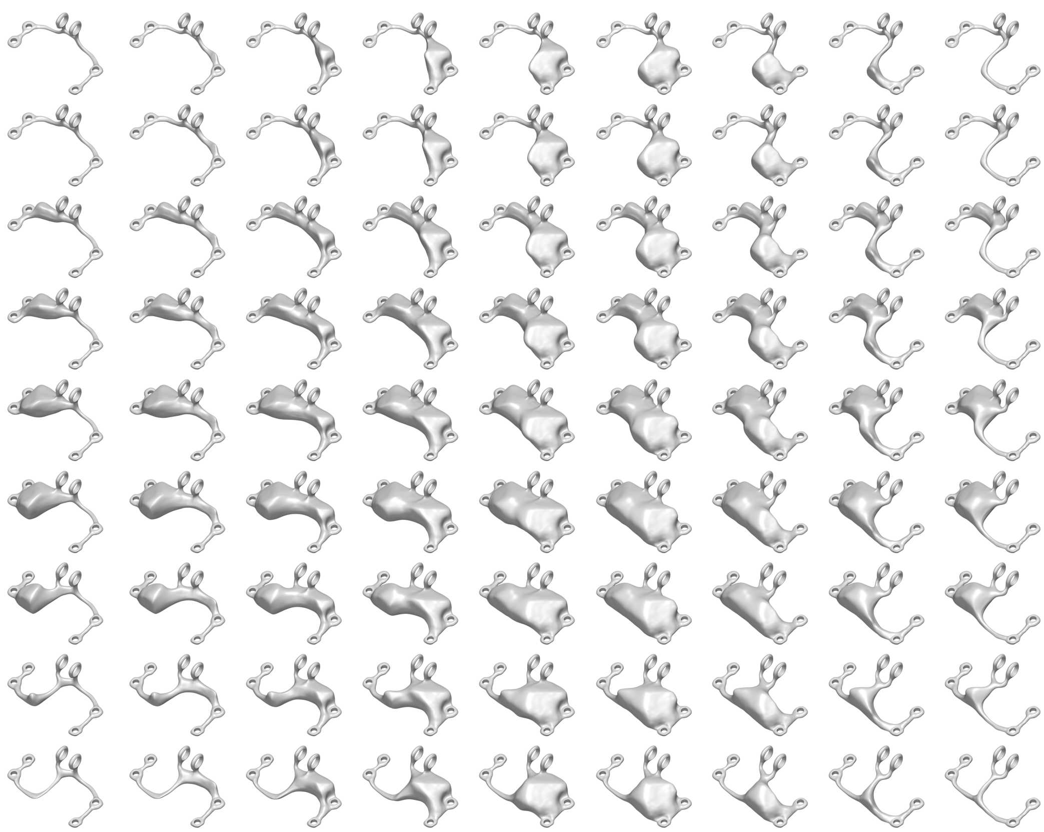

Figure 9: GINNs produce diverse shapes with a structured latent space. The shapes morph continiously into one another when traversing the 2 dimensional latent space. These shapes are produced by the same model as Figure 6. A trained GINN allows the user to sample densely in the latent space with shapes all meeting the constraints: Interfaces are modeled correctly, shapes are not disconnected or leave the design space.

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:

$$L_u = \int_{\mathcal{D}} (D_u \Delta u - uv^2 + \alpha(1-u))^2 \,\mathrm{d}x \;, \quad L_v = \int_{\mathcal{D}} (D_v \Delta v + uv^2 - (\alpha+\beta)v)^2 \,\mathrm{d}x \;. \tag{6}$$

To avoid trivial (i.e. uniform) solutions, we encourage non-zero gradient with a loss term $-\max(1, \int_{\mathcal{D}} (\nabla u(x))^2 + (\nabla v(x))^2 \,\mathrm{d}x)$. We find that architecture and initialization are critical (see Appendix B.1). 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 4. To the best of our knowledge, this is the first PINN that produces 2D Turing patterns in a data-free setting.

B.1 Experimental details

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.
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. The generative PINNs are trained with Adam for 20000 epochs with a 10^{-3} learning rate taking a few minutes.

C Evaluation

C.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 divergence (Nguyen et al., 2021) to compute the divergence measure between two shapes P and Q. For better interpretability, we take the square root of the common definition of chamfer divergence

$$CD_1(P, Q) = \sqrt{\frac{1}{|Q|} \sum_{x \in Q} \min_{y \in P} ||x - y||_2^2} \tag{7}$$

and, similary, for the two-sided chamfer divergence

$$CD_2(P, Q) = \sqrt{\frac{1}{|Q|} \sum_{x \in Q} \min_{y \in P} ||x - y||_2^2 + \frac{1}{|P|} \sum_{x \in P} \min_{y \in Q} ||x - y||_2^2} \;. \tag{8}$$

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.

- $\frac{\text{vol}(\Omega \setminus \mathcal{E})}{\text{vol}(\mathcal{X} \setminus \mathcal{E})}$: 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.
- $\frac{\text{vol}(\Omega \cap \delta\mathcal{E})}{\text{vol}(\delta\mathcal{E})}$: 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.
- $\frac{\text{vol}(DC(\Omega))}{\text{vol}(\mathcal{E})}$: 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.
- $\frac{\text{vol}(DC(\Omega \cap \mathcal{E}))}{\text{vol}(\mathcal{E})}$: 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:

$$\delta_{\text{mean}}(S) = \left[\frac{1}{N} \sum_{i \in [N]} \left(\frac{1}{N-1} \sum_{j \neq i \in [N]} CD_2(\Omega_i, \Omega_j) \right)^{\frac{1}{2}} \right]^2 . \tag{9}$$

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 Section 4). 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 $\kappa_{\text{max}} = 1000000$.

$$E_{\text{strain}}(\Omega) = \frac{1}{N} \sum_{i \in [N]} \min \left[\text{div}^2 \left(\nabla \frac{f(x_i)}{|f(x)|} \right), \kappa_{\text{max}} \right] \tag{10}$$

C.2 JET ENGINE BRACKET

Eikonal loss facilitates learning. Column 2 in Table 3 ablates the eikonal loss. Using the eikonal loss has a positive influence on the connectedness (see $b_0(\omega)$) and provides additional smoothness (see $E_{\text{strain}}(\Omega)$).

Connectedness loss is crucial for connected shapes. Column 3 in Table 3 ablates the connectedness loss. 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. Furthermore, the share of connected interfaces $\frac{CI(\Omega,\mathcal{I})}{n_{\mathcal{I}}}$ is only 0.83. Since for this problem there are 6 interfaces to connect, a value of 0.83 implies that one of the interfaces is disconnected from the others.

Ablation	eikonal	connectedness	smoothness	log-smoothness	base
smoothness	E_{strain}	E_{strain}	E_{strain}	E_{log}	E_{strain}
λ_{eikonal}	0	1	1	1	1
$\lambda_{\text{connectedness}}$	1	0	1	1	1
$\lambda_{\text{smoothness}}$	1	1	0	1	1
$\downarrow b_0(\Omega)$	9	2	1	1	1
$\downarrow b_0(\Omega \cap E)$	7	2	1	1	1
$\downarrow \frac{\text{vol}(DC(\Omega))}{\text{vol}(E)}$	0.00	0.00	0.00	0.00	0.00
$\downarrow \frac{\text{vol}(\Omega \setminus E)}{\text{vol}(X \setminus E)}$	0.00	0.00	0.00	0.00	0.00
$\uparrow \frac{CI(\Omega,I)}{n_{\mathcal{I}}}$	0.67	0.83	1.00	1.00	1.00
$\downarrow CD_1(\Omega, I)$	0.00	0.00	0.00	0.00	0.00
$\downarrow \frac{\text{vol}(\Omega \cap \delta E)}{\text{vol}(\delta E)}$	0.00	0.00	0.00	0.00	0.00
$\downarrow \frac{\text{vol}(DC(\Omega \cap E))}{\text{vol}(E)}$	0.00	0.00	0.00	0.00	0.00
$\downarrow E_{\text{strain}}(\Omega)$	28,115	22,708	8,497	47,842	7,422

Table 3: Metrics for GINNS trained to produce a single shape of the jet engine bracket dataset.

Explicit smoothness also quantitatively improves smoothness. Comparing Table 3, col. 4 to col. 6 shows that not using the smoothness loss leads to less smooth shapes. Qualitatively this is also depicted in Figure 3.

Explicit diversity loss improves diversity. Comparing Table 4, col. 2 to col. 4 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 C.1) is computed using shapes *at the zero level set* of those fields with the chamfer-divergence 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.

Sampling generalizes better than fixed z. Comparing Table 4, col. 3 to col. 4, the metrics for connectedness (e.g. $b_0(\Omega)$) and the number of connected interfaces improve when uniformly sampling the z from the domain during training.

D OPTIMIZATION

In general, an equality-constrained optimization problem can be written as

$$\min_{\theta} O(\theta) \qquad \text{such that} \qquad C_i(\theta) = 0 \quad \forall i \in 0, \dots, m \tag{11}$$

Ablation	diversity	fixed z	base
Training shapes	9	9	9
dim(z)	2	2	2
z sample method	uniform	fix	uniform
smoothness	E_{strain}	E_{strain}	E_{strain}
λ_{eikonal}	1	1	1
$\lambda_{\text{connectedness}}$	1	1	1
$\lambda_{\text{smoothness}}$	1	1	1
λ_{div}	0	1	1
$\downarrow b_0(\Omega)$	1.00	3.00	1.11
$\downarrow b_0(\Omega \cap E)$	1.00	3.00	1.11
$\downarrow \frac{\text{vol}(DC(\Omega))}{\text{vol}(E)}$	0.00	0.00	0.00
$\downarrow \frac{\text{vol}(\Omega \setminus E)}{\text{vol}(X \setminus E)}$	0.00	0.00	0.00
$\uparrow \frac{CI(\Omega, I)}{n_{\mathcal{I}}}$	1.00	0.69	0.96
$\downarrow CD_1(\Omega, I)$	0.00	0.01	0.00
$\downarrow \frac{\text{vol}(\Omega \cap \delta E)}{\text{vol}(\delta E)}$	0.00	0.01	0.00
$\downarrow \frac{\text{vol}(DC(\Omega \cap E))}{\text{vol}(E)}$	0.00	0.01	0.00
$\downarrow E_{\text{strain}}(\Omega)$	23,059	27,563	31,592
$\uparrow \delta_{\text{mean}}$	0.01	0.23	0.21

Table 4: Metrics for GINNS trained to produce multiple shapes of the jet engine bracket dataset. These are aggregated metrics averaged across all shapes.

where $O, C_1 \dots C_m$ are smooth scalar functions $\mathbb{R}^N \to \mathbb{R}$. O is the *objective function* and *constraint functions* C_i represent the collection of equality constraints. A naive approach to solve this optimization problem is to simply relax the constraints into the objective function and solve the unconstrained optimization problem

$$\min_\theta O(\theta) + \mu_{0_k} \sum_{i=0}^{m} C_i(\theta) \tag{12}$$

for a sequence $\{\mu_{0_k}\}$ with $\mu_{0_k} \leq \mu_{0_{k+1}}$ for all k and $\mu_{0_k} \to \infty$. This so-called *penalty method* can however suffer from numerical instabilities for large μ_{0_k}, hence the sequence is generally capped at a maximum value μ_{max}. A further problem, which has recently been studied in regard to PINNs, is that the different objectives in 12 behave on different scales leading to instabilities in training as the gradients of the larger objective functions dominate training.

This issue is addressed by weighting each constraint term individually

$$\min_\theta O(\theta) + \sum_{i=0}^{m} \mu_{i_k} C_i^2(\theta). \tag{13}$$

Besides manual tuning of the weights μ_{i_k}, several schemes to dynamically balance the different terms throughout training have been proposed, such as loss-balancing via the sub-gradients (Wang et al. (2021)), via the eigenvalues of the Neural Tangent Kernel (Wang et al. (2022)) or using a Soft Attention mechanism (McClenny & Braga-Neto (2020)).

A different method for solving 11 is the augmented Lagrangian method (ALM) defined as:

$$\min_{\theta} \max_{\lambda,\mu} \mathcal{L}(\theta, \lambda, \mu) := O(\theta) + \sum_{i=0}^{m} \lambda_i C_i(\theta) + \frac{1}{2}\mu_0 \sum_{i=0}^{m} C_i^2(\theta) \tag{14}$$

Using the min-max inequality or weak duality

$$\max_{\lambda,\mu} \min_{\theta} \mathcal{L}(\theta, \lambda, \mu) \leq \min_{\theta} \max_{\lambda,\mu} \mathcal{L}(\theta, \lambda, \mu) \tag{15}$$

we can solve the max-min problem instead. In each epoch k, a minimization over network parameters θ_k is performed using gradient descent, yielding new parameters θ_{k+1}. Then, the Lagrange multipliers are updated as follows:

$$\lambda_{i_{k+1}} = \lambda_{i_k} + \mu_{0_k} C_i(\theta_{k+1}) \qquad \forall i \in 0, \ldots, m. \tag{16}$$

Note that this so-called dual update of the Lagrange multipliers is simply a gradient ascent step with learning rate μ_{0_k} for each multiplier λ_{i_k}. Typically, there is also an increase of μ_{0_k} up to maximum value μ_{max} as in the penalty method. Constrained optimization with neural networks using the ALM has been shown to perform well in previous works, such as in Son et al. (2023), Kotary & Fioretto (2024), Sangalli et al. (2021), Fioretto et al. (2021) and Basir & Senocak (2023).

In this classical ALM formulation, there is only a single penalty parameter μ_0, which is monotonically increased during optimization. As outlined above, this is often insufficient to handle diverse constraints with different scales. Thus, we opt for the adaptive ALM proposed in Basir & Senocak (2023) using adaptive penalty parameters for each constraint, solving 11 as the unconstrained optimization problem:

$$\max_{\lambda} \min_{\theta} \mathcal{L}(\theta, \lambda, \mu) := o(\theta) + \sum_{i=0}^{m} \lambda_i C_i(\theta) + \frac{1}{2} \sum_{i=0}^{m} \mu_i C_i^2(\theta) \tag{17}$$

In each epoch k again a minimization step over the parameters θ_k via gradient descent is performed. Then the penalty parameters μ_{i_k}, which are simultaneously the learning rate of the Lagrange multipliers λ_{i_k}, are updated using RMSprop followed by the gradient ascent step for λ_{i_k}

$$\bar{\nu}_{i_{k+1}} \leftarrow \alpha \bar{\nu}_{i_k} + (1-\alpha) C_i^2(\theta_{k+1}) \tag{18}$$

$$\mu_{i_{k+1}} \leftarrow \frac{\gamma}{\sqrt{\bar{\nu}_{i_k}} + \epsilon} \tag{19}$$

$$\lambda_{i_{k+1}} \leftarrow \lambda_{i_k} + \mu_{i_k} C_i(\theta_{k+1}) \tag{20}$$

where $\bar{\nu}_i$ is the weighted moving average of the squared gradient w.r.t. λ_i, α is the discounting factor for old gradients, γ is a global learning rate and ϵ is a constant added for the numerical stability of the division. This adaptive approach enables us to handle the diverse set of constraints in GINNs without the need for manual hyperparameter tuning.

Algorithm 1 shows the full algorithm used to train for $\mathcal{T}$ epochs and specifies the hyperparameters we used. The only difference to Basir & Senocak (2023) is we set $\alpha = 0.90$, which is the default value of RMSprop in PyTorch, instead of $\alpha = 0.99$.

Algorithm 1 Adaptive augmented Lagrangian method

Parameters: $\gamma = 1 \times 10^{-2}$, $\alpha = 0.90$, $\epsilon = 1 \times 10^{-8}$
Input: θ_0
Initialize: $\lambda_{0,i} \leftarrow 1$, $\mu_{0,i} \leftarrow 1$, $\overline{v}_{0,i} \leftarrow 0 \; \forall i$
for $t \leftarrow 1$ to $\mathcal{T}$ **do**
    $\theta_t \leftarrow \text{argmin}_\theta \, \mathcal{L}(\theta_{t-1}; \lambda_{t-1}, \mu_{t-1})$ ▷ primal update: a gradient descent step over θ
    $\overline{v}_{t,i} \leftarrow \alpha \overline{v}_{t-1,i} + (1-\alpha) C_i(\theta_t)^2 \; \forall i$
    $\mu_{t,i} \leftarrow \frac{\gamma}{\sqrt{\overline{v}_{t,i}+\epsilon}} \; \forall i$ ▷ penalty update
    $\lambda_{t,i} \leftarrow \lambda_{t-1,i} + \mu_{t,i} C_i(\theta_t) \; \forall i$ ▷ dual update
end for
Output: θ_t

D.1 Loss plots

In Figures 10 and 11 we show the loss plots for training single and multiple shapes respectively. As expected the unweighted losses (middle rows in the Figures) decrease, while the Lagrange terms (bottom rows) increase over training.

E Connectedness

We provide additional details on our approach to the connectedness loss. We break this down in three parts: First, we define the signed distance function of a shape Ω which the neural field we train approximates. Then, we give a short rundown on computing the persistent homology (PH), in particular the PH of a neural field in a not rectangular region. Lastly, we explain how to obtain a differentiable loss on the field from the outputs of the, in general non-differentiable, PH computation.

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

$$f(x) = \begin{cases} d(x, \partial\Omega) & \text{if } x \in \Omega^c \text{ (if } x \text{ is outside the shape)}, \\ -d(x, \partial\Omega) & \text{if } x \in \Omega \text{ (if } x \text{ is inside the shape)}. \end{cases} \tag{21}$$

A point $x \in X$ 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. In INS, the SDF is approximated by a NN with parameters θ: $f_\theta \approx f$.

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, connectedness is an important constraint for enabling GINNs. 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).

E.1 Persistent Homology

Persistent Homology is one of the primary tools which has emerged from topological data analysis (TDA) to extract topological features from data. Data modalities such as point clouds, time series, graphs and n-dimensional images can all be transformed into weighted cell complexes from which the homology can be computed. The homology provides global information about the underlying data and is generally robust.

Homology is an invariant originating from algebraic topology. A topological space X is encoded as cell complexes $C_n(X)$ consisting of n-dimensional balls B^n ($n = 0, 1, 2, ...$) and boundary maps ∂_n from dimension n to $n-1$ which satisfy $\partial_n \circ \partial_{n+1} = 0$ and $\partial_0 = 0$. The homology $H_n(X)$ is then defined as the quotient space

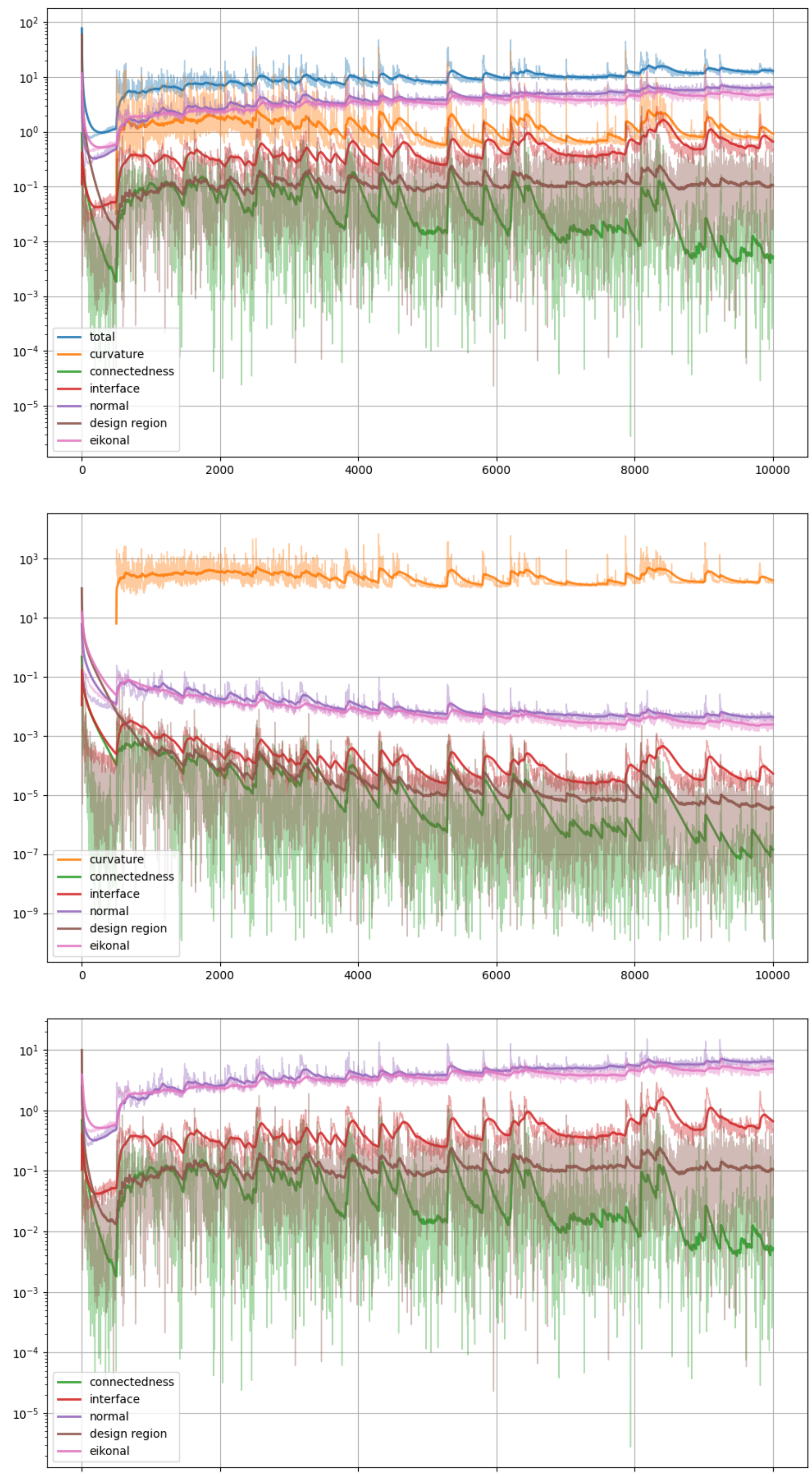

Figure 10: Loss plots for training a *single* shape. The lines with higher alpha are exponential-moved averages of the lower-alpha values by the factor 0.99. (a) The top plot shows the losses as used for backpropagation. (b) The middle plot shows the unweighted losses for individual constraints. (c) The bottom plots show the λ values for the individual constraints.

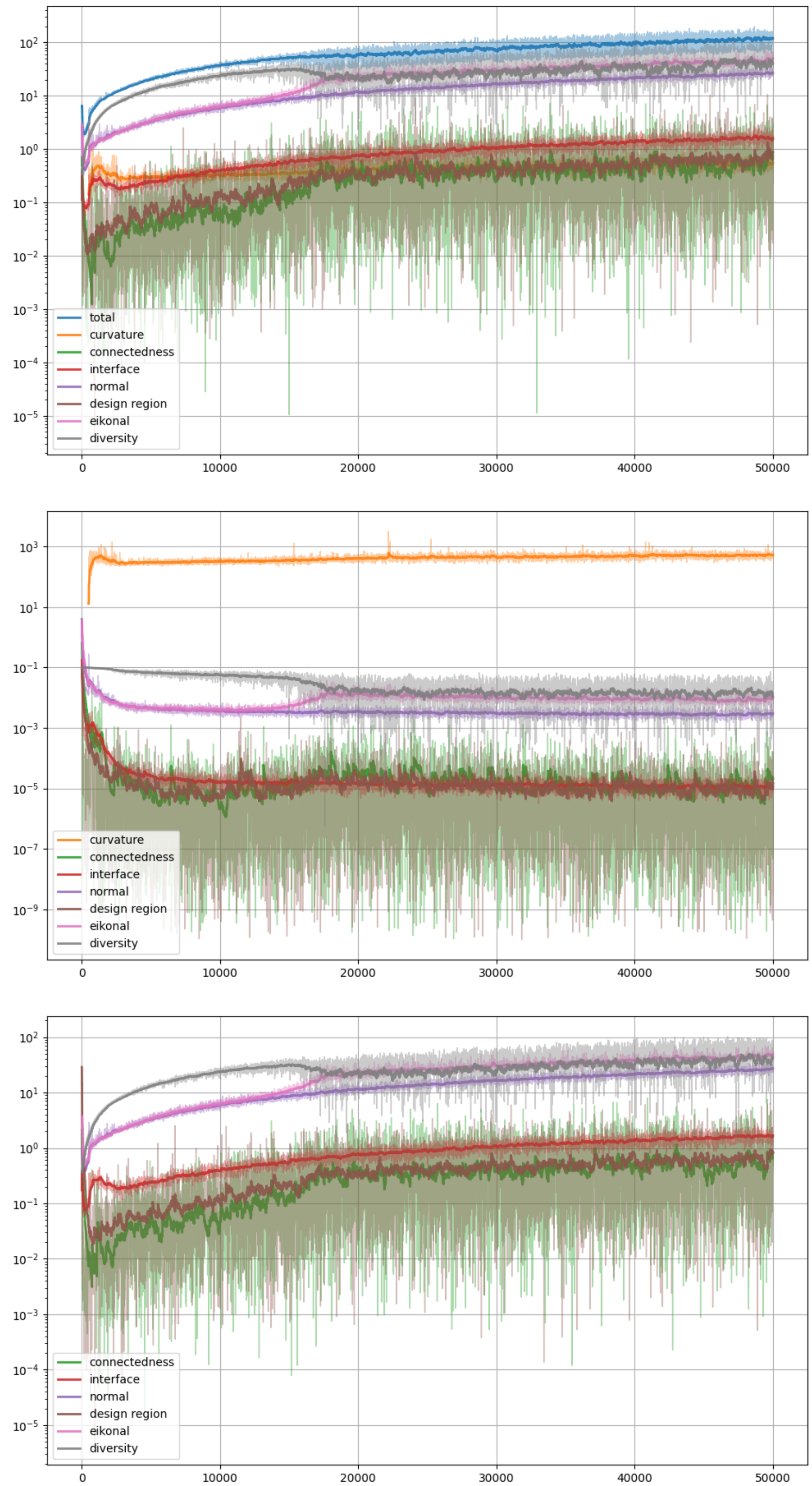

Figure 11: Loss plots for training *multiple* shapes. The lines with lower transparency are exponential-moved averages with factor 0.99 of the higher-transparency values. (a) The top plot shows the losses as used for backpropagation. (b) The middle plot shows the unweighted losses for individual constraints. (c) The bottom plots show the λ values for the individual constraints.

$$H_n(X) = \frac{\ker(\partial_n)}{\mathrm{im}(\partial_{n+1})} \tag{22}$$

The dimension of $H_n(X)$ counts the number of n-dimensional features and defines the Betti number b_n: for $n = 0$ the number of connected components, for $n = 1$ the number of holes, for $n = 2$ the number of voids.

Filtrations on the space X are defined using a filter function $f : X \to \mathbb{R}$. Using a sequence of increasing parameters α_n with $\alpha_k < \alpha_n$ for $k < n$ we can define a sequence of nested subspaces of X as sub-level sets $X_n = f^{-1}([-\infty, \alpha_n])$. We then have

$$\varnothing \subseteq X_1 \subseteq \cdots \subseteq X_N = X \tag{23}$$

The homology of each of these nested complexes $C_n(X_i)$ can be computed.

Persistent Homology encodes how the homology of an increasing sequence of complexes changes under a given filtration. Topological features appear and and vanish as the filter function sweeps over X. The *birth time* b of a feature defined as the value α_n at which the homology of $C_n(X_n)$ changes to include this feature. The *death time* d of a feature is analogously defined as the value α_n at which it is removed from $C_n(X_n)$. The *persistence* of a feature is defined as the length of its lifetime $l = d - b$.

For each Betti number b_n (for each homology class H_n) the information about the persistent homology of a given filtration is encoded in a persistence diagram containing the points (b, d) of the birth and death pairs of all n-dimensional topological features (changes in the dimension of H_n). The persistence diagrams contain the entire topological information about underlying the space or shape for a sufficiently fine filtration.

To compute the persistent homology of a neural field, we evaluate the network on a cubical complex on the domain of the field, i.e. a grid in $\mathbb{R}^N$. The output is simply a gray scale image (we are only dealing with scalar fields in this work) and the PH can computed with existing algorithms. The current SotA algorithm for PH computation on cubical complexes is CRipser Kaji et al. (2020).

Given a grayscale image and a filtration value a, the *sublevel set* at a is the binary image resulting from thresholding the image for values smaller or equal to a. For every such binary image, which defines a weighted cubical complex with coefficients in $\mathbf{Z}/2\mathbf{Z}$, the homology can be computed. The persistence homology is then obtained by sweeping the thresholding value a through $\mathbb{R}$.

In general, we are interested in computing the PH within a given envelope, which is not necessarily a rectangular region. We achieve this by sampling the field in a rectangular domain containing the envelope and setting the value of points not in the envelope to ∞. Applying the PH computation to this altered image then correctly returns the evolution of persistence features within the envelope. The only drawback of this method is the additional computational cost of having to include the grid points outside the envelope in the PH computation, which is why the bounding domain should be chosen tightly around the envelope.

The PH computation itself does not have to be differentiable (and the CRipser library we use is not) because the cells, i.e. the grid point of the image, at which a given persistence feature is born or killed are stored. Hence we can simply use the network output at this grid coordinate to compute the loss and there are no issues concerning differentiability or having to re-implement the PH computation into PyTorch.

E.2 Differentiable topology loss

To compute a differentiable loss, we use the outputs of the PH computation: For each homology class H_n we obtain the points ${}^n p_i$ in the persistence diagram with the associated birth and death times ${}^n b_i$, ${}^n d_i$ and the coordinates of these births $x_{{}^n b_i}$, $y_{{}^n b_i}$, $z_{{}^n b_i}$ and deaths $x_{{}^n d_i}$, $y_{{}^n d_i}$, $z_{{}^n d_i}$.
Remark: The representatives of homology class are not uniquely determined. The CRipser library internally chooses a representative and then outputs its coordinates. In practice this caused no issues.

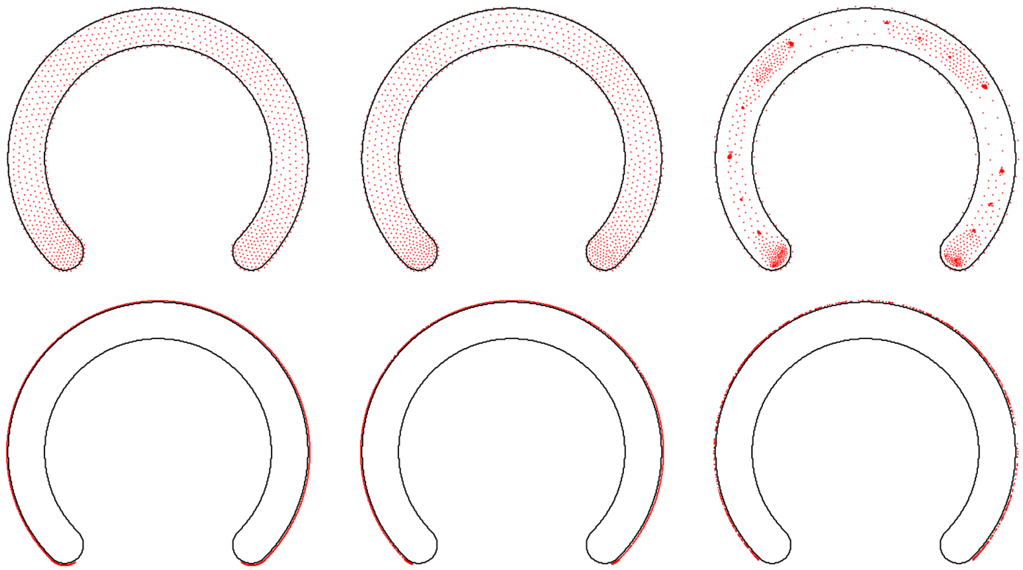

Figure 12: A visual comparison of different diversity losses in a simple 2D example ($\mathcal{F} = \mathbb{R}^2$ and the feasible set $\mathcal{K}$ is the partial annulus). Each point $f \in \mathcal{F}$ represents a candidate solution. The points are optimized to maximize the diversity within the feasible set. The top row shows the *minimal aggregation* δ_{min} as defined in Equation 26. The bottom row shows the *total aggregation* δ_{sum} as defined in Equation 27. 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.

For a selected iso-level a_0 we select all ${}^n p_i$ for which ${}^n b_i < a_0 < {}^n d_i$ and sort them by lifetime ${}^n l_i = {}^n d_i - {}^n b_i$. Now let the index i run from $1 \ldots M$ sorting the selected ${}^n p_i$. To train the network f_θ to produce a single connected component at iso-level a_0 the loss is given by the residuals of the deaths ${}^n d_i$ to a_0 for all $i = 2 \ldots M$, effectively pushing down all but the most persistent component.

$$\mathcal{L}_{cc} = \sum_{i=2}^{M} \Big(a_0 - f_\theta(x_{{}^0 d_i}, y_{{}^0 d_i}, z_{{}^0 d_i}) \Big)^2 \tag{24}$$

It is immediately clear that this term is differentiable with respect to θ.

More generally, to obtain a shape with a Betti number $b_n = m$ at iso-level a_n, the summation above runs from $i = m + 1 \ldots M$. The full topology loss for a N-dimensional shape is then given as

$$\mathcal{L}_{topo} = \sum_{n=0}^{N-1} \sum_{i=m+1}^{M} \big(a_n - f_\theta(x_{{}^n d_i}, y_{{}^n d_i}, z_{{}^n d_i}) \big)^2 \tag{25}$$

F 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 12, 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: $S = \{f_i \in \mathcal{X}, i = 1, \ldots, N\}$. 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

$$\delta_{\text{min}}(S) = \left(\sum_i \left(\min_{j \neq i} d_2(f_i, f_j) \right)^p \right)^{1/p}, \tag{26}$$

as well as the total aggregation measure

$$\delta_{\text{sum}}(S) = \left(\sum_i \left(\sum_j d_2(f_i, f_j) \right)^p \right)^{1/p}. \tag{27}$$

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 always pushes the samples to the extremes of the domain.

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^p distance can also be split into four terms. Using $f(x) = 0 \; \forall x \in \partial\Omega$ we obtain

$$\begin{aligned}
d_2^p(f_i, f_j) &= \int_{\mathcal{X}} (f_i(x) - f_j(x))^p \, \mathrm{d}x \\
&= \int_{\partial\Omega_i \setminus \partial\Omega_j} (0 - f_j(x))^p \, \mathrm{d}x + \int_{\partial\Omega_j \setminus \partial\Omega_i} (f_i(x) - 0)^p \, \mathrm{d}x \\
&+ \int_{\partial\Omega_i \cap \partial\Omega_j} (0 - 0)^p \, \mathrm{d}x + \int_{\mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)} (f_i(x) - f_j(x))^p \, \mathrm{d}x \\
&= \int_{\partial\Omega_i \setminus \partial\Omega_j} f_j(x)^p \, \mathrm{d}x + \int_{\partial\Omega_j \setminus \partial\Omega_i} f_i(x)^p \, \mathrm{d}x + \int_{\mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)} (f_i(x) - f_j(x))^p \, \mathrm{d}x \\
&= \int_{\partial\Omega_i} f_j(x)^p \, \mathrm{d}x + \int_{\partial\Omega_j} f_i(x)^p \, \mathrm{d}x + \int_{\mathcal{X} \setminus (\partial\Omega_i \cup \partial\Omega_j)} (f_i(x) - f_j(x))^p \, \mathrm{d}x
\end{aligned} \tag{28}$$

G Geometric constraints

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

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.
Euler characteristic	Topological constraint.

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