Supplementary material for RESHAPE: Representation Learning for the Explainability of Shapes

Rami Al-Belmpeisi[1,2] (ralbe@dtu.dk), Kristoffer Knutsen Wickstrøm[3], Josefine Vilsbøll Sundgaard[1,2], Peter Hjørringgaard Larsen[2], Anders Bjorholm Dahl[1,†], Robert Jenssen[3,4,5,†], and Michael Kampffmeyer[3,4,†]

[1] Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark
[2] Novo Nordisk A/S, Bagsværd, Denmark
[3] Department of Physics and Technology, UiT The Arctic University of Norway, Tromsø, Norway
[4] Norwegian Computing Center, Oslo, Norway
[5] Pioneer Centre for AI, University of Copenhagen, Copenhagen, Denmark

[†]These authors contributed equally to the supervision of this work.

A Framework implementation

RESHAPE can be understood through the following high-level pseudocode algorithm:

Algorithm 1 RESHAPE framework

```
training_shapes, validation_shapes, test_shapes ← load_data()
model ← 'DeepSDF' or 'DALS'
trained_model ← train(model, training_shapes, validation_shapes)
test_latent_codes ← infer_test_latents(trained_model, test_shapes)
importances ← RESHAPE(test_latent_codes, test_shapes)
```

The detailed code implementation is publicly available in the code repository linked in the main paper.

B Training details

B.1 DeepSDF

We use the original DeepSDF architecture, through a publicly available implementation in [1]. The decoder network f_Θ, which is an 8-layer fully connected Multi-Layer Perceptron (MLP), deviates from the original publication: we use 256 (instead of 512) hidden units per layer, and 64-D latent codes (instead of 512-D), due to GPU memory limitations.

DeepSDF relies on the auto-decoder paradigm, meaning there is no explicit encoder network. Instead, the model maps a continuous 3D coordinate $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ and a shape-specific latent code $\mathbf{z} \in \mathbb{R}^{64}$ to a predicted signed distance value:

$$f_\Theta(\mathbf{x}, \mathbf{z}) \mapsto \mathrm{SDF}(\mathbf{x} \mid \mathbf{z}).$$

Each of the N_{train} training shapes is linked to a learnable latent code $\mathbf{z}_i$ during training. These latent codes are optimised together with the network parameters Θ during training. This involves minimising a truncated L_1 data term for K sampled 3D points $\mathbf{x}_{ij}$, and their ground-truth SDF values s_{ij} for each shape i. For a single point, the loss $\mathcal{L}_{ij}$ is defined as follows:

$$\mathcal{L}_{ij}(\Theta, \mathbf{z}_i) = \big|\, \mathrm{clamp}\big(f_\Theta(\mathbf{x}_{ij}, \mathbf{z}_i), \delta\big) - \mathrm{clamp}(s_{ij}, \delta)\big|,$$

where both predicted, and ground-truth SDFs are restricted to the range $[-\delta, \delta]$ through $\mathrm{clamp}(u, \delta) = \max(-\delta, \min(u, \delta))$. This truncation prevents the loss from being disproportionately dominated by far-field samples.

The latent codes are constrained to a zero-mean distribution via an L_2 regularisation term. Training tunes both the network parameters Θ and per-shape latent codes $\{\mathbf{z}_i\}_{i=1}^{N_{\text{train}}}$ to their optimal values Θ^* and $\{\mathbf{z}_i^*\}_{i=1}^{N_{\text{train}}}$, which are given by minimizing:

$$\min_{\Theta, \{\mathbf{z}_i\}} \sum_{i=1}^{N_{\text{train}}} \left(\sum_{j=1}^{K} \mathcal{L}_{ij}(\Theta, \mathbf{z}_i) + \frac{1}{\sigma^2} \|\mathbf{z}_i\|_2^2 \right)$$

During inference on an unseen test shape, the network parameters are frozen at their trained values Θ^*, while the per-shape latent codes are optimised through the same objective as during training. This test-time optimisation scheme yields the optimal latent code $\mathbf{z}$ that best fits the test shapes under optimisation constraints.

B.2 DALS Implementation Details

In our DALS setup, connected shapes are reconstructed by modelling them as deformations of a fixed template. We use a four-times-subdivided icosahedron of radius $R = 0.1$ as our template mesh, denoted as $\mathcal{T} = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V}$ contains the vertex coordinates $\mathbf{v}_j \in \mathbb{R}^3$ and $\mathcal{F}$ represents the faces.

We deviate from the standard DALS formalism by optimising global (per-shape rather than per-vertex) latent representation along with a Graph Convolutional (GC) decoder g_Θ (instead of MLP). For a vertex j on shape i, the initial input features $\mathbf{h}_{ij}^{(0)}$ are formed by concatenating the 128-D shared latent code $\mathbf{z}_i$ with the template vertex coordinates $\mathbf{v}_j$:

$$\mathbf{h}_{ij}^{(0)} = \big[\mathbf{v}_j; \mathbf{z}_i\big] \in \mathbb{R}^{3+128}$$

The decoder comprises graph-convolutional blocks that utilise LayerNorm (LN) and ReLU activations. The hidden features at layer ℓ are therefore computed as:

$$\mathbf{h}_{ij}^{(\ell)} = \mathrm{ReLU}\Big(\mathrm{LN}\Big(\mathrm{GC}\big(\mathbf{h}_{i\cdot}^{(\ell-1)}, \mathcal{E}\big)_j\Big)\Big), \quad \ell \in \{1,2,3,4\}$$

We use 724, 724, and 362 neurons in the hidden layers, followed by a final GC layer that predicts the 3D vertex offsets. Therefore, the decoder f_Θ predicts a vertex-wise deformation field $\Delta\mathbf{v}_{ij} \in \mathbb{R}^3$, which is conditioned on the per-shape latent code $\mathbf{z}_i$ and template mesh vertex coordinates. The reconstructed mesh has vertex coordinates $\hat{\mathbf{v}}_{ij}$, which are obtained post-deformation through:

$$f_\Theta(\mathbf{z}_i, \mathbf{v}_j) \mapsto \Delta\mathbf{v}_{ij}, \qquad \hat{\mathbf{v}}_{ij} = \mathbf{v}_j + \Delta\mathbf{v}_{ij}$$

This process generates the reconstructed mesh $\hat{\mathcal{R}}_i = (\hat{\mathcal{V}}_i, \mathcal{F})$ for each shape i, through a learned transformation from the template $\mathcal{T}$ to the target shape, where $\hat{\mathcal{V}}_i$ contains the transformed vertex coordinates $\hat{\mathbf{v}}_{ij} \in \mathbb{R}^3$.

During training, we sample B surface points and corresponding unit normals from both the reconstructed mesh ($R_i = \{r_k\}_{k=1}^B$, normals N_i^R) and the target mesh ($T_i = \{t_k\}_{k=1}^B$, normals N_i^T) using farthest point sampling. The optimization minimizes a weighted sum of geometric losses alongside latent regularisation:

$$\min_{\Theta, \{\mathbf{z}_i\}} \sum_{i=1}^{N_{\text{train}}} \Big[\lambda_{\text{ch}}\mathcal{L}_{\text{ch}}^{(i)} + \lambda_{\text{n}}\mathcal{L}_{\text{n}}^{(i)} + \lambda_{\text{e}}\mathcal{L}_{\text{e}}^{(i)} + \lambda_{\text{q}}\mathcal{L}_{\text{q}}^{(i)} + \frac{1}{\sigma^2}\|\mathbf{z}_i\|_2^2\Big] \tag{1}$$

The individual loss components are defined as follows:

- **Chamfer Distance (point-to-point):** Enforces proximity between reconstructed and target meshes.

$$\mathcal{L}_{\text{ch}}^{(i)}(\hat{\mathcal{R}}_i, T_i) = \mathcal{D}_{CD}(\hat{\mathcal{R}}_i, T_i) = \frac{1}{|R_i|}\sum_{r\in R_i}\min_{t\in T_i}\|r-t\|_2^2 + \frac{1}{|T_i|}\sum_{t\in T_i}\min_{r\in R_i}\|t-r\|_2^2 \tag{2}$$

- **Normal Consistency:** Penalizes disagreement between normals of closely corresponding points, where $t^* = \arg\min_{t\in T_i}\|r-t\|_2$ and $r^* = \arg\min_{r\in R_i}\|t-r\|_2$.

$$\mathcal{L}_{\text{n}}^{(i)}(\hat{\mathcal{R}}_i, T_i) = \frac{1}{|R_i|}\sum_{r\in R_i}\Big(1 - \langle N_r,\, N_{t^*}\rangle\Big) + \frac{1}{|T_i|}\sum_{t\in T_i}\Big(1 - \langle N_t,\, N_{r^*}\rangle\Big) \tag{3}$$

- **Edge Length Regularization:** Promotes consistent edge lengths for the reconstructed edge set $\mathcal{E}_i$, where $\bar{\ell}$ is the mean edge length and $\ell_e = \|\hat{\mathbf{v}}_u - \hat{\mathbf{v}}_v\|$.

$$\mathcal{L}_{\text{e}}^{(i)} = \frac{1}{|\mathcal{E}_i|}\sum_{e\in\mathcal{E}_i}\big(\ell_e - \bar{\ell}\big)^2 \tag{4}$$

– **Triangle Quality:** Penalizes low-quality, degenerate triangles. For a triangular face $f = (a, b, c) \in \mathcal{F}$ with area A_f and quality metric $q(f) = (4\sqrt{3}\,A_f)/(\ell_{ab}^2 + \ell_{bc}^2 + \ell_{ca}^2)$, the loss is:

$$\mathcal{L}_{\mathrm{q}}^{(i)} = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \big(1 - q(f)\big) \tag{5}$$

In our setup, the empirical loss weights are set to $\lambda_{\mathrm{ch}} = 1$, $\lambda_{\mathrm{n}} = 0.01$, $\lambda_{\mathrm{e}} = 0.01$, $\lambda_{\mathrm{q}} = 0.001$, and the latent prior variance is $\sigma = 0.1$.

As in DeepSDF, inference on an unseen test shape fixes the decoder parameters at their optimised values, Θ^*. We optimise only the latent code $\mathbf{z}_i$, which is initialised from the average training latent code, by minimising the same geometric objectives:

$$\arg \min_{\{\mathbf{z}_i\}_{i=1}^{N_{\mathrm{test}}}} \sum_{i=1}^{N_{\mathrm{test}}} \Big[\lambda_{\mathrm{ch}}\,\mathcal{L}_{\mathrm{ch}}^{(i)} + \lambda_{\mathrm{n}}\,\mathcal{L}_{\mathrm{n}}^{(i)} + \lambda_{\mathrm{e}}\,\mathcal{L}_{\mathrm{e}}^{(i)} + \lambda_{\mathrm{q}}\,\mathcal{L}_{\mathrm{q}}^{(i)} + \frac{1}{\sigma^2}\,\|\mathbf{z}_i\|_2^2\Big] \tag{6}$$

C Additional details

C.1 Mask to mesh conversion

The medical meshes are derived by applying the marching cubes algorithm on the volumetric masks. However, due to voxel anisotropy, this yields meshes that suffer from staircase artefacts. A dedicated post-processing pipeline is used to obtain clinically usable surfaces. The pipeline utilises VTK [2] functionality to repair connectivity and merge duplicate or near-duplicate vertices, and to convert fragmented primitives into a coherent manifold-like surface. While this step reduces staircase artefacts, smoothing is then required, which is achieved by using a low-shrinkage windowed sinc Taubin filter with 20 iterations and a passband of 0.01. This achieves smooth, artefact-free meshes across all medical geometries.

C.2 Point Sampling density estimates

Determining point sampling density across all experiments is essential for balancing computational efficiency and geometric fidelity. Since the mesh surfaces are 2-dimensional manifolds, the expected scaling of the distance between independent point sets is $d \propto N^{-1/2}$. This naturally follows from the area of a Voronoi cell associated with each point, where $A_{cell} \approx A_{total}/N$ and the scaling of 2D surfaces $A_{cell} \propto d^2$, which suggests that $d \propto N^{-1/2}$. Using 10 synthetic data surfaces, and $10^3 - 10^6$ points oversampling using Farthest Point Sampling, to achieve near-equidistant distribution, we verify this scaling, and extract exact fit parameter (see Fig. 1). Reaching a desired precision of $\mathcal{D}_{CD} = 10^{-2}$ would then require about $N \approx 6,340$ sampled points per mesh.

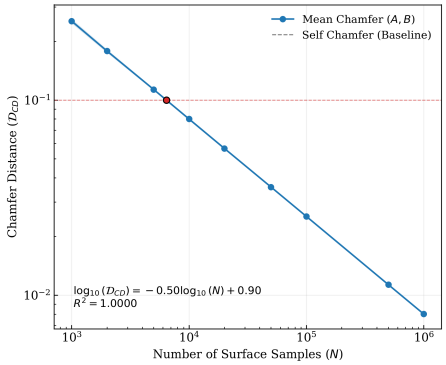

Fig. 1. Scaling of the Chamfer distance $\mathcal{D}_{CD}$ as a function of the number of sampled points N. The experimental data confirm the theoretical scaling law of $N^{-1/2}$. The fitted line provides an analytical expression for determining the required sampling density to achieve a desired geometric precision.

C.3 Training hyperparameters

For the synthetic shapes and the DALS model, we used $M = 2,000$ stochastic masks with a masking probability of $P = 0.5$. However, a less aggressive masking strategy was necessary for the DeepSDF model, due to positive-only SDF predictions often produced as a result of excessive perturbation of the DeepSDF latent code. The inability to extract the zero-SDF isocontour in these situations leads to undefined similarity scores. To circumvent this, we use $P = 0.9$. Moreover, a 64^3 grid to resolve the $(x, y, z) \in [-1, 1]^3$ domain across all RESHAPE implementations on DeepSDF.

References

1. Comi, M.: Deepsdf-minimal. `https://github.com/maurock/DeepSDF/` (2023)
2. Schroeder, W., Martin, K., Lorensen, B.: The Visualization Toolkit (4th ed.). Kitware (2006)