SAL: Sign Agnostic Learning of Shapes from Raw Data

Matan Atzmon and Yaron Lipman
Weizmann Institute of Science
Rehovot, Israel

1. Implementation details

1.1. Surface reconstruction

Data Preparation. For each point cloud $\mathcal{X} \subset \mathbb{R}^3$ we created training data by sampling two Gaussian variables centered at each $\boldsymbol{x} \in \mathcal{X}$. We set the standard deviation of the first Gaussian to be the distance to the 50-th closest point in $\mathcal{X}$, whereas the second Gaussian standard deviation is set to be the distance to the furthest point in $\mathcal{X}$.

Network Architecture. We use an MLP as detailed in equation 8 and equation 9 with $\ell = 7$, *i.e.*, 8 layers, where $d_i^{out}, d_i^{in} = 512$ for interior layers and $d_1^{in} = 3$. We also add a skip connection at the 4-th layer concatenating the input point $\boldsymbol{x} \in \mathbb{R}^3$ with the hidden variable $\boldsymbol{y} \in \mathbb{R}^{509}$, *i.e.*, $[\boldsymbol{x}, \boldsymbol{y}] \in \mathbb{R}^{512}$, where accordingly $d_3^{out} = 509$.

Training Details. We train the network in the surface reconstruction experiment with ADAM optimizer [3], learning rate 0.0001 for 5000 epochs. Training was done on an Nvidia V-100 GPU, using PYTORCH deep learning framework [6].

1.2. Learning shape space

Data Preparation. To speed up training on the D-Faust dataset, we preprocessed the original scans $\mathcal{X} \subset \mathbb{R}^3$ and sampled 500K points from each scan. The size of the sample is similar to [5]. First, we sampled 250K points uniformly (w.r.t. area) from the triangle soup $\mathcal{X}$ and then placed two Gaussian random variables centered at each sample point $\boldsymbol{x} \in \mathcal{X}$. We set the standard deviation of the first Gaussian to be the distance to the 50-th closest point in $\mathcal{X}$, whereas the second Gaussian standard deviation is set to be 0.2. For each sample point we calculated its unsigned distance to the closest triangle in the triangle soup. The distance computation was done using [1].

Network Architecture. Our network architecture is Encoder-Decoder based, where for the encoder we have used PointNet [7] and DeepSets [8] layers. Each layer is composed of

$$\begin{aligned} \mathrm{PFC}(d_{\text{in}}, d_{\text{out}}) &: \boldsymbol{X} \mapsto \nu\left(\boldsymbol{X}W + \mathbf{1}b^T\right) \\ \mathrm{PL}(d_{\text{in}}, 2d_{\text{in}}) &: \boldsymbol{Y} \mapsto [\boldsymbol{Y}, \max(\boldsymbol{Y})\mathbf{1}] \end{aligned}$$

where $[\cdot,\cdot]$ is the concat operation. Our Architecture is

$$\begin{aligned} &\mathrm{PFC}(3,128) \to \mathrm{PFC}(128,128) \to \mathrm{PL}(128,256) \to \\ &\mathrm{PFC}(256,128) \to \mathrm{PL}(128,256) \to \mathrm{PFC}(256,128) \to \\ &\mathrm{PL}(128,256) \to \mathrm{PFC}(256,128) \to \mathrm{PL}(128,256) \to \\ &\mathrm{PFC}(256,256) \to \mathrm{MaxPool} \to \mathrm{FC}(256,256), \end{aligned}$$

similarly to [4] SimplePointNet architecture. For the decoder, our network architecture is the same as in the Surface reconstruction experiment except for $d_1^{in} = 256 + 3$, as the decoder receives as input $[z, x] \in \mathbb{R}^{259}$. Where $[z, x]$ is a concatenation of the latent encoding $\boldsymbol{z} \in \mathbb{R}^{256}$ and a point $\boldsymbol{x} \in \mathbb{R}^3$ in space.

Training Details. Training our networks for learning the shape space of the D-Faust dataset was done with the following choices. We have used the ADAM optimizer [3], initialized with learning rate 0.0005 and batch size of 64. We scheduled the learning rate to decrease every 500 epochs by a factor of 0.5. We stopped the training process after 2000 epochs. Training was done on 4 Nvidia V-100 GPUs, using PYTORCH deep learning framework [6].

1.3. Additional Experiments

Single reconstruction versus VAE reconstruction. One of the key advantages of SAL is that it can be used for reconstructing a surface from a single input scan or incorporated into a VAE architecture for learning a shape space from an entire scans dataset. This raises an interesting question, whether learning a shape space has also an impact on the quality of the reconstructions. To answer this question, we ran SAL surface reconstruction on each of the scans used for training the main experiment of the paper (See table 2 for more details). When comparing our SAL VAE

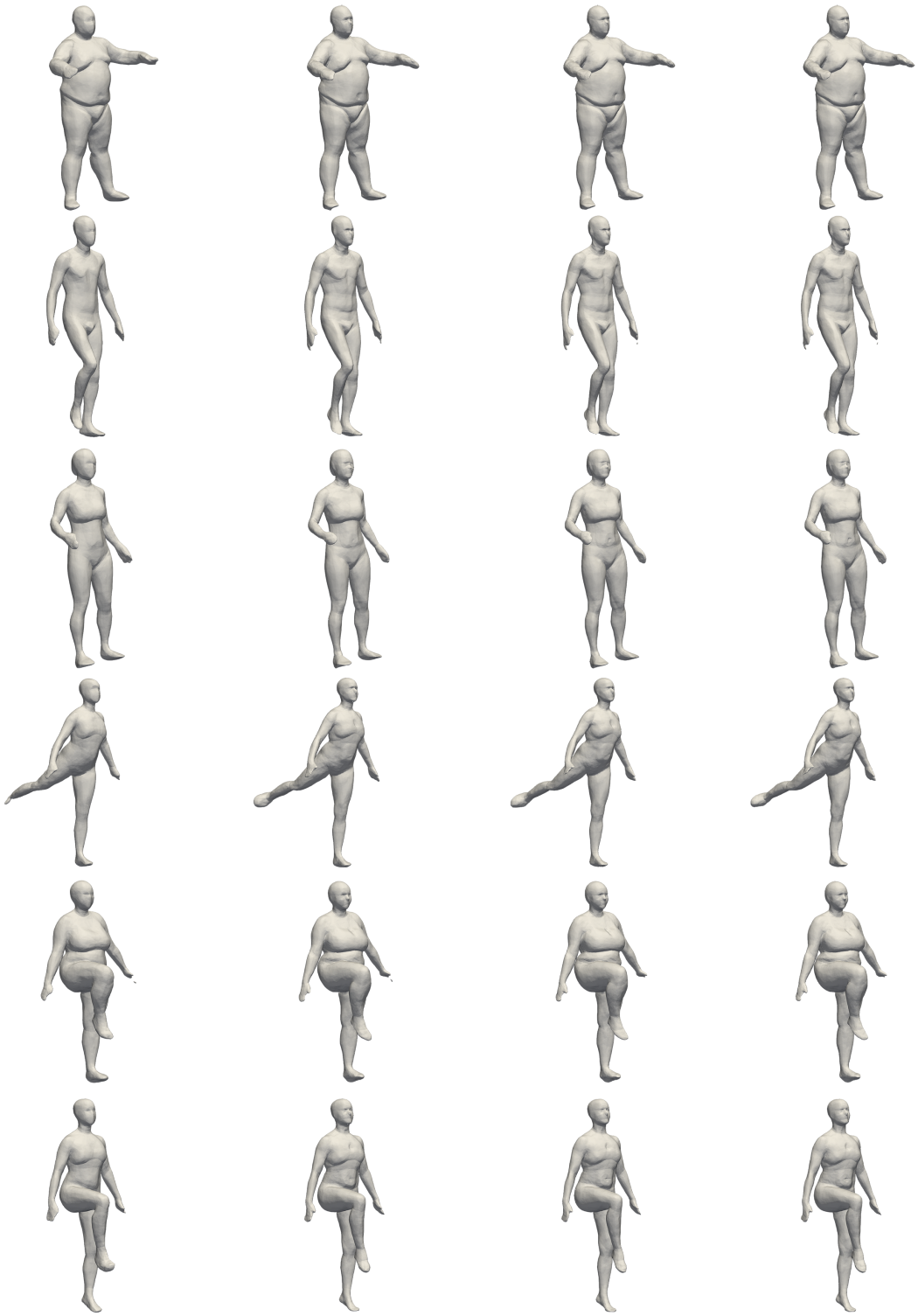

Figure 1: Effect of the number of training epochs on reconstruction quality of test scans. Left to right in each row: epoch 500, 1500, 2500 and 3500.

training results on the registrations (ground truth) versus SAL single reconstruction we see differences in favor of our VAE learner, whereas the results on the original scans are comparable. That is, SAL single reconstruction results are $0.10, 0.17, 0.22; 0.07, 0.08, 0.10$ on the registrations and scans for the $5\%, 50\%, 95\%$ percentiles respectively.

Number of epochs used for training SAL VAE. Figure 1 shows reconstructions of test scans for different stages of training on the D-Faust dataset. Given the main paper discussion on SAL limit signed function, we additionally add reconstructions from relatively advanced epoch as 3500, showing that no error in contouring occur.

2. Proofs

2.1. Proof of Theorem 3

Theorem 3. *Consider a linear model $f(\boldsymbol{x};\boldsymbol{\theta}) = \varphi(\boldsymbol{w}^T\boldsymbol{x} + b)$, $\boldsymbol{\theta} = (\boldsymbol{w}, b)$, with a strong non-linearity $\varphi : \mathbb{R} \to \mathbb{R}$. Assume the data $\mathcal{X}$ lies on a plane $\mathcal{P} = \{\boldsymbol{x} | \boldsymbol{n}^T\boldsymbol{x} + c = 0\}$,* i.e., *$\mathcal{X} \subset \mathcal{P}$. Then, there exists $\alpha \in \mathbb{R}_+$ so that $(\boldsymbol{w}^*, b^*) = (\alpha\boldsymbol{n}, \alpha c)$ is a critical point of the loss in equation 2.*

Proof. For simplicity, we restrict our attention to absolutely continuous measures $D_{\mathcal{X}}$, that is defined by a continuous density function $\mu(\boldsymbol{x})$. Generalizing to measures with a discrete part (such as the one we use for the L^0 distance, for example) can be proven similarly.

Denoting $\boldsymbol{\theta} = (\boldsymbol{w}, b)$, the loss in equation 2 can be written as

$$\text{loss}(\boldsymbol{w}, b) = \int_{\mathbb{R}^d} \tau(f(\boldsymbol{x};\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{x}))\mu(\boldsymbol{x})d\boldsymbol{x}.$$

Denote by $\boldsymbol{r} : \mathbb{R}^d \to \mathbb{R}^d$ the linear reflection w.r.t. $\mathcal{P}$, *i.e.*, $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{x} - 2\boldsymbol{n}\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{x}_0) = (I - 2\boldsymbol{n}\boldsymbol{n}^T)\boldsymbol{x} - 2c\boldsymbol{n}$, where $\boldsymbol{x}_0 \in \mathcal{P}$ is an arbitrary point. $h_{\mathcal{X}}$ and μ are invariant to $\boldsymbol{r}$, that is $h_{\mathcal{X}}(\boldsymbol{r}(\boldsymbol{x})) = h_{\mathcal{X}}(\boldsymbol{x})$, $\mu(\boldsymbol{r}(\boldsymbol{x})) = \mu(\boldsymbol{x})$.

The gradient of the loss is $\nabla_{\boldsymbol{w},b}\text{loss}(\boldsymbol{w}, b) =$

$$\int_{\mathbb{R}^d} \frac{\partial\tau}{\partial a}(f(\boldsymbol{x};\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{x}))\nabla_{\boldsymbol{w},b}f(\boldsymbol{x};\boldsymbol{\theta})\mu(\boldsymbol{x})d\boldsymbol{x}, \quad (1)$$

where $\nabla_{\boldsymbol{w},b}f(\boldsymbol{x};\boldsymbol{\theta}) = \varphi'(\boldsymbol{w}^T\boldsymbol{x} + b)[\boldsymbol{x}, 1]$.

Let $\boldsymbol{w} = \alpha\boldsymbol{n}$ and $b = \alpha c$, where $\alpha \in \mathbb{R}_+$ is currently arbitrary. We decompose $\mathbb{R}^d$ to the two sides of $\mathcal{P}$, $\Omega_+ = \{\boldsymbol{x} | \boldsymbol{n}^T\boldsymbol{x} + c \geq 0\}$, and $\Omega_- = \mathbb{R}^d \setminus \Omega_+$. Now consider the integral in equation 1 restricted to Ω_- and perform change of variables $\boldsymbol{x} = \boldsymbol{r}(\boldsymbol{y})$; note that $\boldsymbol{r}$ consists of an orthogonal linear part and therefore $d\boldsymbol{y} = \left|\det \frac{\partial\boldsymbol{y}}{\partial\boldsymbol{x}}\right| d\boldsymbol{x} = d\boldsymbol{x}$. Furthermore, property (i) implies that $\frac{\partial\tau}{\partial a}(a, b) = -\frac{\partial\tau}{\partial a}(-a, b)$, and since φ is anti-symmetric and $\boldsymbol{w}^T\boldsymbol{r}(\boldsymbol{x}) + b = -(\boldsymbol{w}^T\boldsymbol{y} + b)$ we get

$$\frac{\partial\tau}{\partial a}(f(\boldsymbol{r}(\boldsymbol{y});\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{r}(\boldsymbol{y}))) = -\frac{\partial\tau}{\partial a}(f(\boldsymbol{y};\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{y})).$$

As φ is anti-symmetric, φ' is symmetric, *i.e.*, $\varphi'(a) = \varphi'(-a)$ and therefore

$$\varphi'(\boldsymbol{w}^T\boldsymbol{r}(\boldsymbol{y}) + b) = \varphi'(\boldsymbol{w}^T\boldsymbol{y} + b).$$

Plugging these in the integral after the change of variables we reach

$$-\int_{\Omega_+} \frac{\partial\tau}{\partial a}(f(\boldsymbol{y};\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{y}))\varphi'(\boldsymbol{w}^T\boldsymbol{y} + b)[\boldsymbol{r}(\boldsymbol{y}), 1]\mu(\boldsymbol{y})d\boldsymbol{y}.$$

An immediate consequence is that $\nabla_b\text{loss}(\boldsymbol{w}, b) = 0$. As for $\nabla_{\boldsymbol{w}}\text{loss}(\boldsymbol{w}, b)$ we have:

$$\int_{\Omega_+} \frac{\partial\tau}{\partial a}(f(\boldsymbol{x};\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{x}))\varphi'(\boldsymbol{w}^T\boldsymbol{x} + b)(\boldsymbol{x} - \boldsymbol{r}(\boldsymbol{x}))\mu(\boldsymbol{x})d\boldsymbol{x}.$$

Since $\boldsymbol{x} - \boldsymbol{r}(\boldsymbol{x}) = 2\boldsymbol{n}\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{x}_0)$ we get $\nabla_{\boldsymbol{w}}\text{loss}(\boldsymbol{w}, b) =$

$$2\boldsymbol{n}\int_{\Omega_+} \frac{\partial\tau}{\partial a}(f(\boldsymbol{x};\boldsymbol{\theta}), h_{\mathcal{X}}(\boldsymbol{x}))\varphi'(\boldsymbol{w}^T\boldsymbol{x}+b)\boldsymbol{n}^T(\boldsymbol{x}-\boldsymbol{x}_0)\mu(\boldsymbol{x})d\boldsymbol{x}.$$

The last integral is scalar and we denote its integrand by $g_\alpha(\boldsymbol{x})$ (remember that $(\boldsymbol{w}, b) = \alpha(\boldsymbol{n}, c)$), *i.e.*,

$$\nabla_{\boldsymbol{w}}\text{loss}(\boldsymbol{w}, b) = 2\boldsymbol{n}\int_{\Omega_+} g_\alpha(\boldsymbol{x})d\boldsymbol{x} = 2\boldsymbol{n}G(\alpha),$$

where $G(\alpha) = \int_{\Omega_+} g_\alpha(\boldsymbol{x})d\boldsymbol{x}$. The proof will be done if we show there exists $\alpha \in \mathbb{R}_+$ so that $G(\alpha) = 0$. We will use the intermediate value theorem for continuous functions to prove this, that is, we will show existence of $\alpha_-, \alpha_+ \in \mathbb{R}_+$ so that $G(\alpha_-) < 0 < G(\alpha_+)$.

Let us show the existence of α_+. Let $\boldsymbol{x} \in \Omega_+$ such that $\gamma = \boldsymbol{n}^T\boldsymbol{x} + c > 0$ and $\mu(\boldsymbol{x}) > 0$. Then,

$$g_\alpha(\boldsymbol{x}) = \frac{\partial\tau}{\partial a}(\varphi(\alpha\gamma), h_{\mathcal{X}}(\boldsymbol{x}))\varphi'(\alpha\gamma)\gamma\mu(\boldsymbol{x}).$$

Furthermore, since $\beta^{-1} \geq \varphi'(a) \geq \beta > 0$ we have that

$$g_\alpha(\boldsymbol{x}) \geq \frac{\partial\tau}{\partial a}(\varphi(\alpha\gamma), h_{\mathcal{X}}(\boldsymbol{x}))\gamma\mu(\boldsymbol{x}) \begin{cases} \beta & \frac{\partial\tau}{\partial a} \geq 0 \\ \beta^{-1} & \frac{\partial\tau}{\partial a} < 0 \end{cases}$$
$$= \tilde{g}_\alpha(\boldsymbol{x}).$$

Note that $\tilde{g}_\alpha(\boldsymbol{x})$ is monotonically increasing and for sufficiently large $\alpha \in \mathbb{R}_+$ we have that $\tilde{g}_\alpha(\boldsymbol{x}) > 0$. Since $\int_{\Omega_+} \tilde{g}_\alpha(\boldsymbol{x})d\boldsymbol{x} > -\infty$ for all $\alpha \in \mathbb{R}_+$ the integral monotone convergence theorem implies that $\lim_{\alpha\to\infty} \int_{\Omega_+} \tilde{g}_\alpha(\boldsymbol{x})d\boldsymbol{x} > 0$. Lastly, since

$$\int_{\Omega_+} g_\alpha(\boldsymbol{x})d\boldsymbol{x} \geq \int_{\Omega_+} \tilde{g}_\alpha(\boldsymbol{x})d\boldsymbol{x},$$

the existence of α_+ is established. The case of α_- is proven similarly. □

2.2. Local plane reproduction with MLP

Theorem 1. *Consider an MLP as defined in equation 8. Assume that locally in some domain $\Omega \subset \mathbb{R}^d$ the data $\mathcal{X} \cap \Omega$ lies on a plane $\mathcal{P} = \{\boldsymbol{x} | \boldsymbol{n}^T \boldsymbol{x} + c = 0\}$, i.e., $\mathcal{X} \cap \Omega \subset \mathcal{P}$. Then, there exists a critical point $\boldsymbol{\theta}^*$ of the loss in equation 2 that reconstructs $\mathcal{P}$ locally in Ω.*

By "reconstructs $\mathcal{P}$ locally" we mean that $\boldsymbol{\theta}^*$ is critical for the loss if $D_{\mathcal{X}}$ is sufficiently concentrated around any point in $\mathcal{P} \cap \Omega$.

Proof. We next consider a general MLP model $f(\boldsymbol{x}; \boldsymbol{\theta})$ as in equation 8. We denote by $\text{supp}(\mu)$ the support set of μ, *i.e.*, $\{\boldsymbol{x} | \mu(\boldsymbol{x}) > 0\}$. Let us write the layers of the network $f(\boldsymbol{x}; \boldsymbol{\theta})$ using only matrix multiplication, that is

$$f_i(\boldsymbol{y}) = \text{diag}\big(H(\boldsymbol{W}_i \boldsymbol{y} + \boldsymbol{b}_i)\big)\big(\boldsymbol{W}_i \boldsymbol{y} + \boldsymbol{b}_i\big), \qquad (2)$$

where $H(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases}$ is the Heaviside function.

Let μ be so that $\text{supp}(\mu) \subset \Omega$, and fix an arbitrary $\boldsymbol{x}_0 \in \text{supp}(\mu) \cap \mathcal{P}$. Next, let $\boldsymbol{\theta} \in \mathbb{R}^m$ be such that: (a) there exists a domain Υ, so that $\text{supp}(\mu) \subset \Upsilon$ and for which all the diagonal matrices in equation 2 are constants, *i.e.*, Υ is a domain over which $f(\boldsymbol{x}; \boldsymbol{\theta})$ (excluding the final non-linearity φ) is linear as a function of $\boldsymbol{x}$. Therefore, for $\boldsymbol{x} \in \Upsilon$ we have the layers of f satisfy

$$f_i(\boldsymbol{y}) = D_i\big(\boldsymbol{W}_i \boldsymbol{y} + \boldsymbol{b}_i\big),$$

and D_i are constant diagonal matrices. Over this domain we have

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \varphi\big(\boldsymbol{w}(\boldsymbol{\theta})^T \boldsymbol{x} + b(\boldsymbol{\theta})\big),$$

and therefore

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{A} \nabla_{\boldsymbol{w},b} f(\boldsymbol{x}; \boldsymbol{\theta}), \qquad (3)$$

where $\nabla_{\boldsymbol{w},b} f(\boldsymbol{x}; \boldsymbol{\theta}) = \varphi'(\boldsymbol{w}(\boldsymbol{\theta})^T \boldsymbol{x} + b(\boldsymbol{\theta}))[\boldsymbol{x}, 1]$, and $\boldsymbol{A} \in \mathbb{R}^{m \times (d+1)}$ is a matrix holding the partial derivatives of $\boldsymbol{w}(\boldsymbol{\theta})$ and $b(\boldsymbol{\theta})$. (b) over Υ there exists $\boldsymbol{w}(\boldsymbol{\theta}) = \boldsymbol{n}$ and $b(\boldsymbol{\theta}) = c$. The existence of such $\boldsymbol{\theta}$ is guaranteed for example from Theorem 2.1 in [2].

Similarly to the proof of Theorem 3 there exists $\alpha \in \mathbb{R}_+$ so that $\nabla_{\boldsymbol{w},b} \text{loss}(\alpha \boldsymbol{w}(\boldsymbol{\theta}), \alpha b(\boldsymbol{\theta})) = 0$. Scaling the $\boldsymbol{w}, b$ components of $\boldsymbol{\theta}$ by α and denoting the new parameter vector by $\boldsymbol{\theta}^*$, we get that $\nabla_{\boldsymbol{w},b} \text{loss}(\boldsymbol{w}(\boldsymbol{\theta}^*), b(\boldsymbol{\theta}^*)) = 0$ (see equation 1) and therefore equation 3 implies that $\nabla_{\boldsymbol{\theta}} \text{loss}(\boldsymbol{\theta}^*) = 0$, as required. □

2.3. Proof of Theorem 1

Theorem 1. *Let f be an MLP (see equations 8-9). Set, for $1 \leq i \leq \ell$, $\boldsymbol{b}_i = 0$ and $\boldsymbol{W}_i$ i.i.d. from a normal distribution $\mathcal{N}(0, \frac{\sqrt{2}}{\sqrt{d_i^{out}}})$; further set $\boldsymbol{w} = \frac{\sqrt{\pi}}{\sqrt{d_\ell^{out}}} \mathbf{1}$, $c = -r$. Then, $f(\boldsymbol{x}) \approx \varphi(\|x\| - r)$.*

Proof. To prove this theorem reduce the problem to a single hidden layer network, see Theorem 2. Denote $g(\boldsymbol{x}) = f_{\ell-1} \circ \cdots \circ f_1$. If we prove that $\|g(\boldsymbol{x})\| \approx \|\boldsymbol{x}\|$ then Theorem 2 implies the current theorem.

It is enough to consider a single layer: $h(\boldsymbol{x}) = \nu(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$. For brevity let $k = d^{out}$. Now,

$$\|h(\boldsymbol{x})\|^2 = \sum_{i=1}^{k} \nu(\boldsymbol{W}_{i,:} \cdot \boldsymbol{x})^2 = \frac{1}{k} \sum_{i=1}^{k} \nu(\sqrt{k} \boldsymbol{W}_{i,:} \cdot \boldsymbol{x})^2.$$

Note that the entries of $\sqrt{k} \boldsymbol{W}_{i,:}$ are distributed i.i.d. $\mathcal{N}(0, \sqrt{2})$. Hence by the law of large numbers the last term converge to

$$\begin{aligned}
&\|\boldsymbol{x}\|^2 \int_{\mathbb{R}^k} \nu\left(\boldsymbol{y} \cdot \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right)^2 \mu(\boldsymbol{y}) dy \\
&= \|\boldsymbol{x}\|^2 \int_{\mathbb{R}^k} \nu(y_1)^2 \frac{1}{(2\pi\sigma^2)^{k/2}} e^{-\frac{\|\boldsymbol{y}\|}{2\sigma^2}} dy \\
&= \|\boldsymbol{x}\|^2 \int_{\mathbb{R}} \nu(y_1)^2 \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y_1^2}{2\sigma^2}} dy \\
&= \frac{\|\boldsymbol{x}\|^2}{2} \int_{\mathbb{R}} y_1^2 \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y_1^2}{2\sigma^2}} dy \\
&= \|\boldsymbol{x}\|^2,
\end{aligned}$$

where in the second equality, similarly to the proof of Theorem 2, we changed variables, $\boldsymbol{y} = \boldsymbol{R}\boldsymbol{y}'$, where we chose $\boldsymbol{R} \in \mathbb{R}^{k \times k}$ orthogonal so that $\boldsymbol{R}^T \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} = (1, 0, \ldots, 0)^T$. □

Skip connections. Adapting the geometric initialization to skip connections is easy: consider skip connection layers of the form $s(\boldsymbol{y}) = \frac{1}{\sqrt{2}}(\boldsymbol{y}, \boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d$ is the input to the network and $\boldsymbol{y} \in \mathbb{R}^{d_i^{out}}$ is some interior hidden variable. Then $\|s(\boldsymbol{y})\|^2 = \frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{1}{2}\|\boldsymbol{y}\|^2$. According to Theorem 1 we have $\|\boldsymbol{y}\| \approx \|\boldsymbol{x}\|$ and hence $\|s(\boldsymbol{y})\|^2 \approx \|\boldsymbol{x}\|^2$.

2.4. Proof of Theorem 2

Theorem 2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an MLP with ReLU activation, ν, and a single hidden layer. That is, $f(\boldsymbol{x}) = \boldsymbol{w}^T \nu(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + c$, where $\boldsymbol{W} \in \mathbb{R}^{d^{out} \times d}$, $\boldsymbol{b} \in \mathbb{R}^{d^{out}}$, $\boldsymbol{w} \in \mathbb{R}^{d^{out}}$, $c \in \mathbb{R}$ are the learnable parameters. If $\boldsymbol{b} = 0$, $\boldsymbol{w} = \frac{\sqrt{2\pi}}{\sigma d^{out}} \mathbf{1}$, $c = -r$, $r > 0$, and all entries of $\boldsymbol{W}$ are i.i.d. normal $\mathcal{N}(0, \sigma^2)$ then $f(\boldsymbol{x}) \approx \|\boldsymbol{x}\| - r$. That is, f is approximately the signed distance function to a $d-1$ sphere of radius r in $\mathbb{R}^d$, centered at the origin.*

Proof. For brevity we denote $k = d^{out}$. Note that plugging $\boldsymbol{w}, \boldsymbol{b}, c$ in f we get $f(\boldsymbol{x}) = \frac{\sqrt{2\pi}}{\sigma k} \sum_{i=1}^{k} \nu(\boldsymbol{w}_i \cdot \boldsymbol{x}) - r$, where $\boldsymbol{w}_i$ is the i^{th} row of $\boldsymbol{W}$. Let μ denote the density of multivariate normal distribution $\mathcal{N} = (0, \sigma^2 I_k)$. By the law of

large numbers, the first term converges to

$$\begin{aligned}
&\frac{\sqrt{2\pi}}{\sigma}\int_{\mathbb{R}^k}\nu\left(\boldsymbol{u}\cdot\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\|\boldsymbol{x}\|\right)\mu(\boldsymbol{u})d\boldsymbol{u}\\
&=\frac{\sqrt{2\pi}\,\|\boldsymbol{x}\|}{\sigma}\int_{\mathbb{R}^k}\nu\left(\boldsymbol{u}\cdot\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right)\mu(\boldsymbol{u})d\boldsymbol{u}\\
&=\frac{\sqrt{2\pi}\,\|\boldsymbol{x}\|}{\sigma}\int_{\mathbb{R}^k}\nu(v_1)\mu(\boldsymbol{v})d\boldsymbol{v}\\
&=\frac{\sqrt{2\pi}\,\|\boldsymbol{x}\|}{\sigma}\int_{\mathbb{R}^k}\nu(v_1)\frac{1}{(2\pi\sigma^2)^{k/2}}e^{-\frac{\|\boldsymbol{v}\|^2}{2\sigma^2}}d\boldsymbol{v}\\
&=\frac{\sqrt{2\pi}\,\|\boldsymbol{x}\|}{\sigma}\int_{\mathbb{R}}\nu(v_1)\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{v_1^2}{2\sigma^2}}dv_1\\
&=\frac{\sqrt{2\pi}\,\|\boldsymbol{x}\|}{2\sigma}\int_{\mathbb{R}}|v_1|\,\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{v_1^2}{2\sigma^2}}dv_1\\
&=\|\boldsymbol{x}\|\,,
\end{aligned}$$

where in the second equality we changed variables, $\boldsymbol{u} = \boldsymbol{R}\boldsymbol{v}$, where we chose $\boldsymbol{R} \in \mathbb{R}^{k\times k}$ orthogonal so that $\boldsymbol{R}^T\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} = (1,0,\ldots,0)^T$, and used the rotation invariance of μ, namely $\mu(\boldsymbol{R}\boldsymbol{v}) = \mu(\boldsymbol{v})$. In the last equality we used the mean of the folded normal distribution. Therefore we get $f(\boldsymbol{x}) \approx \|\boldsymbol{x}\| - r$. □

References

[1] Pierre Alliez, Simon Giraudot, Clément Jamin, Florent Lafarge, Quentin Mérigot, Jocelyn Meyron, Laurent Saboret, Nader Salman, and Shihao Wu. Point set processing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. 1

[2] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016. 4

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[4] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1

[5] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1

[7] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 1

[8] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 1