A Learnable Radial Basis Positional Embedding for Coordinate-MLPs

Sameera Ramasinghe [1] **and Simon Lucey** [2]

[1]Amazon, [2]University of Adelaide
ramasisa@amazon.com

Abstract

We propose a novel method to enhance the performance of coordinate-MLPs (also referred to as neural fields) by learning instance-specific positional embeddings. End-to-end optimization of positional embedding parameters along with network weights leads to poor generalization performance. Instead, we develop a generic framework to learn the positional embedding based on the classic graph-Laplacian regularization, which can implicitly balance the trade-off between memorization and generalization. This framework is then used to propose a novel positional embedding scheme, where the hyperparameters are learned per coordinate (*i.e* instance) to deliver optimal performance. We show that the proposed embedding achieves better performance with higher stability compared to the well-established random Fourier features (RFF). Further, we demonstrate that the proposed embedding scheme yields stable gradients, enabling seamless integration into deep architectures as intermediate layers.

Introduction

Encoding continuous signals as weights of neural networks is now becoming a ubiquitous strategy in computer vision due to numerous appealing properties. First, signals in computer vision are typically treated in a discrete manner, *e.g.*, images, voxels, meshes, and point clouds. In contrast, neural representations allow such signals to be treated as continuous over bounded domains, a major virtue being smooth interpolations to unseen coordinates (Niemeyer et al. 2020; Saito et al. 2019; Sitzmann, Zollhöfer, and Wetzstein 2019; Mildenhall et al. 2020; Zhong et al. 2019). Second, neural representations are differentiable and generally more compact compared to canonical representations (Sitzmann et al. 2020; Dupont et al. 2021). For instance, a 3D mesh can be represented using a multi-layer perceptron (MLP) with only a few layers (Takikawa et al. 2021; Yu et al. 2021a). Consequently, this recent influx of neural signal representations has supplemented a variety of computer vision tasks including images representation (Nguyen, Yosinski, and Clune 2015; Stanley 2007), texture generation (Henzler, Mitra, and Ritschel 2020; Oechsle et al. 2019; Henzler, Mitra, and Ritschel 2020; Xiang et al. 2021), shape representation (Chen and Zhang 2019; Deng et al. 2020; Tiwari et al. 2021; Genova et al. 2020; Basher, Sarmad, and Boutellier 2021; Mu et al. 2021; Park et al. 2019), and novel view synthesis (Mildenhall et al. 2020; Niemeyer et al. 2020; Saito et al. 2019; Sitzmann, Zollhöfer, and Wetzstein 2019; Yu et al. 2021b; Pumarola et al. 2021; Rebain et al. 2021; Martin-Brualla et al. 2021; Wang et al. 2021; Park et al. 2021).

Ground truth End-to-end training Proposed training

Figure 1: Optimizing the positional embedding layer by minimizing the training error by updating the embedding hyperparameters and the network weights (end-to-end optimization) leads to overfitting, yielding poor generalization. In contrast, the proposed optimization procedure achieves a better trade-off between memorization and generalization. We use (25%) regularly sampled training points.

The elementary units of neural signal representations are coordinate-MLPs. In particular, a coordinate-MLP consumes a discrete set of coordinates as inputs $\mathbf{x}$ and corresponding targets $\mathbf{y}$, and strives to learn a continuous approximation $f : \mathbb{R}^M \rightarrow \mathbb{R}^N$. However, it has been both empirically (Mildenhall et al. 2020; Niemeyer et al. 2020) and theoretically (Tancik et al. 2020) observed that low-dimensional coordinate inputs (typically 2 or 3 dimensional) are sub-optimal for such tasks, since conventional MLPs, when trained with such inputs, fail to learn functions with high-frequency components. A popular approach to tackle this obstacle involves projecting the coordinates into a higher dimensional space via a *positional embed-*

ding prior to the MLP. Positional embeddings – like MLPs – have parameters that can be tuned to match the signal. Unlike MLPs, however, they cannot be learned end-to-end; when they are, it can lead to catastrophic generalization/interpolation performance (see Figure 1). To circumvent this issue, most methods instead tie the positional embedding parameters across coordinates (dramatically reducing the search space, often to a single parameter), and then select the reduced parameter set through a cross-validation procedure. This inevitably leads to sub-optimal results, as the optimal selection varies from coordinate-to-coordinate and signal-to-signal (Tancik et al. 2020; Zheng, Ramasinghe, and Lucey 2021). Philosophically, this approach is questionable: Modern deep learning has heavily advocated for the learning of parameters (even those in positional embedding) from data. Our paper presents a way forward on how to effectively learn instance (*i.e.* coordinate) specific positional embeddings for coordinate-MLPs.

Why does the optimization of positional embedding parameters in Figure 1 – using gradient descent to minimize the training error – catastrophically fail? We postulate that the root cause of this behavior extends to the inherent trade-off between the memorization (of seen coordinates) and generalization (to unseen coordinates) of the embedding. Unlike modern MLPs, the generalization performance of positional embedding must be explicitly – rather than architecturally – controlled. In this sense, the values of positional encoding should be treated to control the learning process (*i.e.* hyper-parameters) rather than learned through conventional end-to-end training. Our paper proposes that this can be done by optimizing a proxy measure of generalization: namely smoothness. We show that the desired level of smoothness varies locally and equivalently, the distortion of the manifold formed by the positional embedding layer should be smooth with respect to the first-order gradients of the encoded signal. Then, for practical implementation, we extend our analysis to its discrete counter-part using similarity graphs and link the above objective to the graph-Laplacian regularization.

Contributions: We demonstrate the effectiveness of our approach across a number of signal reconstruction tasks in comparison to leading hard coded methods such as Random Fourier Frequencies (RFF) made popular in (Mildenhall et al. 2020). A unique aspect of our method is that we do NOT use Fourier positional embeddings in our approach. Such embeddings are actually problematic when attempting to enforce local smoothness. Inspired by (Zheng, Ramasinghe, and Lucey 2021), we instead opt for embeddings that have spatial locality such as radial basis functions. To our knowledge, this is the first time that: (i) non-Fourier, and (ii) learned hyper-parameters are used to obtain state-of-the-art performance in coordinate-MLPs. We also show that our embedder allows more stable back-propagations through the positional embedding layer, enabling hassle-free integration of the positional embedding layer as an intermediate module in deep architectures. It is also noteworthy that our work sheds light on a rather debatable topic – deep networks can benefit from non-end-to-end learning, where each layer can be optimized independently with different optimization goals. However, a broader exposition of this topic is out of the scope of this paper.

Related Works

Positional embeddings are an integral component of coordinate-MLPs that enable them to encode high-frequency functions with fine details. To this date, random Fourier features (RFF) have been the prevalent method of embedding positions. One of the earliest works that employed RFF for embedding positions extends back to the preliminary work by Rahimi *et al.* (Rahimi, Recht et al. 2007), who found that an arbitrary stationary kernel can be approximated using sinusoidal input mappings. Their work was primarily inspired by Bochner's theorem. Such embeddings recently gained momentum after the seminal work by Mildenhall *et al.* (Mildenhall et al. 2020), where they used random Fourier features to successfully synthesize novel views of a 3D scene. Since then, a large and growing body of literature has investigated the effects of modulating low dimensional inputs with Fourier features (Xu, Alldieck, and Sminchisescu 2021; Jain, Tancik, and Abbeel 2021; Deng et al. 2021; Yen-Chen et al. 2020; Trevithick and Yang 2020; Bi et al. 2020; Guo et al. 2020). These works provide significant heuristic evidence that such pre-processing allows MLPs to regress functions with high-frequency content. A recent follow-up theoretical examination by Tancik *et al.* (Tancik et al. 2020) further confirmed that Fourier embeddings allow tuning the bandwidth of the neural tangent kernel (NTK) of MLPs, enabling them to learn high frequency content. In contrast, Zheng *et al.* (Zheng, Ramasinghe, and Lucey 2021) proposed a novel embedding scheme that involves non-Fourier functions, where they construct the embeddings via samples of shifted basis functions. Despite the efficacy of their method in encoding 1D signals, extending the Gaussian embedder to encode 2D or 3D signals necessitates sampling 2D or 3D Gaussian signals on a dense grid. Thus, the dimension of the embedded coordinates grows exponentially with the dimension of the encoded signal, which is a significant drawback compared to RFF.

Learned embeddings: Positional embeddings have enjoyed data-driven optimizations in language and sequence modeling. For instance, Shaw *et al.* (Shaw, Uszkoreit, and Vaswani 2018) proposed a relative position representation to reduce the number of parameters. Similarly, Lan *et al.* (Lan et al. 2019) and Dehghani *et al.* (Dehghani et al. 2018) proposed to inject relative position information to every layer of a transformer for improved performance. In contrast, Wang *et al.* (Wang et al. 2019) proposed to encode positions by complex numbers of different phases and wave-lengths. A major drawback of their method was that it increases the size of the embedding matrix by a factor of three. Although the aforementioned methods use priors from data to enhance the embeddings, they are not automatically learnable from data. Liu *et al.* (Liu et al. 2020), on the other hand, introduced a learnable positional encoding scheme, where they model the evolution of encoded results along position index by a dynamical system. In order to model this dynamical system, they utilize neural ODEs (Chen et al. 2018). However, all of the above methods focus on embedding word positions when

modeling sequential data. Our work focuses on embedding coordinates to encode continuous signals using MLPs.

Smoothness of Positional Embedding

MLPs naturally preserve smoothness when trained end-to-end to fit a signal (Tancik et al. 2020; Rahaman et al. 2019). That is, they do not tend to critically overfit when encoding functions with high frequencies. On the other hand, positional embeddings demonstrate quite the opposite characteristics by rapidly overfitting to the training inputs when optimized similarly to the MLP weights. Therefore, a crucial aspect that should be considered in any optimization scheme, that attempts to learn positional embeddings, is preserving a form of smoothness with respect to the targets. Consider a simple signal $f(x) : \mathbb{R} \rightarrow \mathbb{R}$. We opt to preserve,

$$\frac{\|\Phi(x + \Delta x) - \Phi(x)\|_2}{|f(x + \Delta x) - f(x)|} = K \tag{1}$$

where $\Phi(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^D$ is the positional embedding. With $\lim_{\Delta x \rightarrow 0}$ we get,

$$\frac{\|d\Phi(x)\|_2}{|dx|} \propto \frac{|df(x)|}{|dx|} \tag{2}$$

Assume that $\Phi(\cdot)$ has nonvanishing Jacobian almost everywhere, so that its image in $\mathbb{R}^D$ is a manifold with Riemannian metric inherited from $\mathbb{R}^D$. Then, $\|d\Phi(x)\|_2$ can be written in terms of the Riemannian metric tensor $\mathcal{M}$ which gives,

$$\frac{|\sqrt{\det(\mathcal{M}(x))}dx|}{|dx|} \propto \frac{|df(x)|}{|dx|}, \tag{3}$$

where $\sqrt{\det(\mathcal{M}(x))}$ is the volume element of $\mathcal{M}$ at x. It follows that,

$$\sqrt{\det(\mathcal{M}(x))} \propto |\frac{df(x)}{dx}|. \tag{4}$$

Multi-dimensional signals: When encoding vector valued functions $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M$, we approximate the above objective as,

$$\sqrt{\det(\mathcal{M}(\mathbf{x}))} \propto \|\mathbf{J}_f(\mathbf{x})\|_F, \tag{5}$$

where $\mathbf{J}_f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of f at $\mathbf{x}$ and $\|\cdot\|_F$ is the Frobenius norm. This portrays that the volume element of the manifold induced by the positional embedding layer should be proportional to the norm of the first-order gradients (Jacobian) of targets. Intuitively, the goal of Eq. 5 can be perceived as follows: $\det(\mathcal{M}(\mathbf{x}))$ is a measure of distortion of a manifold with respect to its coordinate-chart at $\mathbf{x}$. Hence, at intervals where $f(\mathbf{x}_i)$ has high gradients with respect to the coordinates, the manifold will consist of higher distortions. Therefore, the distance between two points on the manifold will be stretched, and the gradients of the outputs with respect to the manifold will be encouraged to remain approximately constant.

However, a critical obstacle that hampers direct utilization of the above analysis is that, in practice, we are only armed with discrete training samples within a bounded space in $\mathbb{R}^N$. Therefore, it is vital to investigate a scheme that can approximate 5 using a discrete representation. Next, we seek to accomplish this task using similarity graphs.

Discrete Approximation of the Manifold

A graph can be considered as a discrete approximation of a manifold (Zhou and Burges 2008). In the *graph signal processing* (GSP) literature, *exemplar functions* are used to lift a low dimensional signal to a higher-dimensional space, which is then used to construct an undirected similarity graph. We observe that positional embedders perform a similar task, *i.e.* project low-dimensional input coordinates to a higher-dimensional space. Following these insights, we construct an undirected similarity graph $\mathcal{G}$ using the positional embeddings with the intention of approximating 5 with a discrete domain formulation.

Consider a set of coordinates $\{\mathbf{x}_i\}_{i=1}^L$ and corresponding sampled outputs $\{f(\mathbf{x}_i)\}_{i=1}^L$, $\mathbf{x} \in \mathbb{R}^N, f(\mathbf{x}) \in \mathbb{R}^M$. Then, with an embedding function $\phi : \mathbb{R}^N \rightarrow \mathbb{R}^D$ defined on a bounded interval in $\mathbb{R}^N$, we obtain a set of $D-$dimensional vectors $\{\Phi(\mathbf{x}_i)\}_{i=1}^L$, where $\Phi(\mathbf{x}_i) = [\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \ldots \phi_D(\mathbf{x}_i)]^T$. Treating this set of vectors $\{\Phi(\mathbf{x}_i)\}_{i=1}^L \in \mathcal{V}$ as vertices, we build an undirected graph $\mathcal{G}$ with edges $\mathcal{E}$, where the edge weight w_{ij} between two vertices $\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \in \mathcal{V}$ is defined as

$$w_{ij} = (\rho_i \rho_j)^{-\lambda} \theta(d_{i,j}), \tag{6}$$

Such that

$$\theta(d_{i,j}) = \exp\left(-\frac{d_{i,j}^2}{2\epsilon^2}\right), \tag{7}$$

where $d_{i,j} = \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|_2$ and $\rho_i = \sum_{j=1}^L \theta(d_{i,j})$. Further, ϵ and λ are hyper-parameters (we fix these parameters across all our experiments, hence, hyperparameter sweeps are not required). The term $(\rho_i \rho_j)^{-\lambda}$ normalizes the edge weights with respect to the degree of the corresponding vertex. Likewise, Eq. 7 smooths out the edge weights while suppressing weights between vertices that are too distant. Finally, the unnormalized graph Laplacian $\mathbf{L}$ of $\mathcal{G}$ can be computed as

$$\mathbf{L} = \mathbf{A} - \mathbf{D}, \tag{8}$$

where $\mathbf{A}$ and $\mathbf{D}$ are the adjacency matrix and the degree matrix of $\mathcal{G}$, respectively. Next, we discuss the impact of Laplacian regularization of $\mathcal{G}$ on our original objective 5.

Graph Laplacian: The regularization objective is defined as

$$\tau(\mathbf{u}) = \mathbf{u}^T \mathbf{L} \mathbf{u}, \tag{9}$$

where $\mathbf{u} = [\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \ldots, \varphi(\mathbf{x}_L)]^T$ and $\varphi : \mathbb{R}^N \rightarrow \mathbb{R}$ is an arbitrary smooth function defined on the coordinate space. Minimizing 9 encourages $w_{i,j}$ to be larger if $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$ are similar (Appendix). As the distance between inputs approaches zero, Eq. 9 converges to its continuous

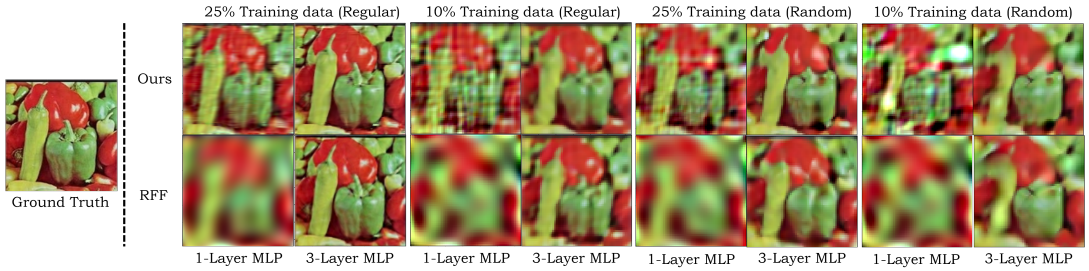

Figure 2: A qualitative comparison of the super-Gaussian embedder against RFF in 2D image representations. The super-Gaussian embedder achieves significantly better results with a linear MLP compared to RFF under various sampling conditions. However, as deeper MLPs with non-linearity are used, RFF representations rapidly increase in quality, although our method consistently produces better results (Table 2). We used 128×128 images for this example for clear visual contrast.

counter-part, *i.e. anistropic Dirichlet energy* (Pang and Cheung 2017). Further, minimizing the anistropic Dirichlet energy is equivalent to enforcing the quantity $\det(\mathbf{G}(\mathbf{x}))$ to be smooth with respect to $\mathbf{u}$ (Alliez et al. 2007) where

$$\mathbf{G}(\mathbf{x}) = \sum_{i=1}^{D} \begin{bmatrix} (\frac{\partial \phi_i(\mathbf{x})}{\partial x_1})^2 & \frac{\partial \phi_i(\mathbf{x})}{\partial x_1}\frac{\partial \phi_i(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial \phi_i(\mathbf{x})}{\partial x_1}\frac{\partial \phi_i(\mathbf{x})}{\partial x_N} \\ \frac{\partial \phi_i(\mathbf{x})}{\partial x_2}\frac{\partial \phi_i(\mathbf{x})}{\partial x_1} & (\frac{\partial \phi_i(\mathbf{x})}{\partial x_2})^2 & \dots & \frac{\partial \phi_i(\mathbf{x})}{\partial x_2}\frac{\partial \phi_i(\mathbf{x})}{\partial x_N} \\ & \vdots & & \\ \frac{\partial \phi_i(\mathbf{x})}{\partial x_N}\frac{\partial \phi_i(\mathbf{x})}{\partial x_1} & \frac{\partial \phi_i(\mathbf{x})}{\partial x_N}\frac{\partial \phi_i(\mathbf{x})}{\partial x_2} & \dots & (\frac{\partial \phi_i(\mathbf{x})}{\partial x_N})^2 \end{bmatrix}$$

Smoothness here means that $\det(\mathbf{G}(\mathbf{x}))$ is higher if $\varphi(\mathbf{x})$ is higher, and vise-versa. On the other hand, we make the following interesting observation. Consider the Jacobian of the embedded manifold

$$\mathbf{J}_\Phi(\mathbf{x}) = \begin{bmatrix} \frac{\partial \phi_1(\mathbf{x})}{\partial x_1} & \frac{\partial \phi_2(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial \phi_D(\mathbf{x})}{\partial x_1} \\ & \vdots & & \\ \frac{\partial \phi_1(\mathbf{x})}{\partial x_N} & \frac{\partial \phi_2(\mathbf{x})}{\partial x_N} & \dots & \frac{\partial \phi_D(\mathbf{x})}{\partial x_N} \end{bmatrix}$$

Then, the Riemannian metric tensor $\mathcal{M}(\mathbf{x}) = \mathbf{J}_\Phi(\mathbf{x})\mathbf{J}_\Phi^T(\mathbf{x})$ of the manifold created by the positional embedding is equivalent to $\mathbf{G}$ given in Eq. . Hence, choosing $\varphi(\mathbf{x}) = \|\mathbf{J}_f(\mathbf{x})\|_F$ and minimizing $\tau(\mathbf{u})$ encourages our embedded manifold to adhere to the constraint 5. This is a crucial observation that motivates our work. That is, constructing a graph using embedded positions as vertices, and applying the Laplacian regularization against the Jacobian norm of the encoded signal approximately fulfills 5.

Radial Basis Embedders

In this section, we propose a novel trainable positional embedding scheme based on the radial basis functions that can benefit from the optimization framework developed thus far. Using the radial basis functions to encode positions were partially explored in (Zheng, Ramasinghe, and Lucey 2021). However, in the original form, their extension to higher dimensions is problematic (see Appendix) and yield inferior performance to RFF. In contrast, the proposed embedders do not suffer from such a drawback. Due to superior empirical performance, we choose a super-Gaussian as our basis function. Next, we define and discuss super-Gaussian embedders.

Definition 1 *Given an N-dimensional input coordinate $\boldsymbol{x} = [x_1, \dots, x_N]^T$, the D-dimensional super-Gaussian positional embedder takes the form:*

$$\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_D(\mathbf{x})]^T,$$

where $phi_i(\boldsymbol{x}) = \left[e^{-\frac{(\boldsymbol{x}\cdot\alpha - t_i)^2}{2\sigma_{\boldsymbol{x}}^2}}\right]^b$. Here, t_i and b are constants, $\alpha \in \mathbb{R}^N$ is a vector with constant elements, and $\sigma_{\boldsymbol{x}}$ is a coordinate-dependant parameter.

Now, we discuss optimizing the super-Gaussian positional embedder using the graph Laplacian regularizer. Precisely, our aim is to obtain a coordinate dependant set of standard deviations $\{\sigma_{\mathbf{x}}\}$, which minimizes 9, and thereby approximates 5. We begin by affirming that the super-Gaussian embedders indeed form a manifold with its local coordinate chart as the input space. Formally, we present the following proposition.

Proposition 1 *Let the positional embedding be $\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x})\, \phi_2(\boldsymbol{x})\, \dots \phi_D(\boldsymbol{x})]^T$ where $\phi_i(\boldsymbol{x}) = \left[e^{\frac{(\boldsymbol{x}\cdot\alpha - t_i)^2}{\sigma^2}}\right]^b$ for $\boldsymbol{x} \in \mathbb{R}^N$, t_i and b are constants, and $\alpha \in \mathbb{R}^N$ is a vector with constant elements. Then, Φ is a $N-$dimensional manifold in an ambient $\mathbb{R}^D$ space.*

(Proof in Appendix). Then, we construct a similarity graph as discussed earlier using the embedded vectors. Afterward, one can optimize $\sigma_{\mathbf{x}}$ to minimize 9 using the gradient descent. However, this can lead to unexpected trivial solutions as discussed next. Let $\mathcal{E}$ be the set of edges of our graph $\mathcal{G}$. It can be shown that,

$$\tau(\mathbf{u}) = \mathbf{u}^T\mathbf{L}\mathbf{u} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j))^2 \qquad (10)$$

(see Appendix). Therefore, if $\tau(\mathbf{u})$ is minimized in an unconstrained setting, the network can converge to a trivial solution by minimizing $w_{i,j}$ without preserving any graph structure. In other words, the network seeks to put the embedded coordinates as further apart as possible by decreasing $\sigma_{\mathbf{x}}$ everywhere. To avoid the above trivial solution, we add a second term to the objective function as,

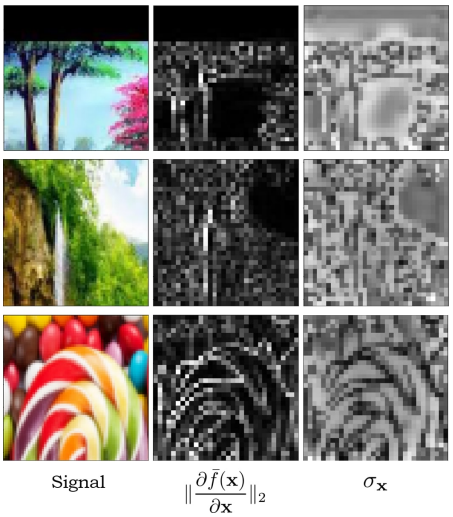

Figure 3: $\|\frac{\partial \bar{f}(\mathbf{x})}{\partial \mathbf{x}}\|_2$ vs the trained standard deviations $\sigma_{\mathbf{x}}$ where $\bar{f}(\mathbf{x})$ is the mean of the encoded signal at $\mathbf{x}$. The standard deviations are *almost* a point-wise property of $\|\frac{\partial \bar{f}(\mathbf{x})}{\partial \mathbf{x}}\|_2$. This enables us to approximate the standard deviations for unseen signals using a polynomial equation without training each time. We use 16×16 patches for this example to clearly visualize the pixel-wise correlations.

$$\bar{\tau}(\mathbf{u}) = \mathbf{u}^T \mathbf{L} \mathbf{u} - \lambda \|\mathbf{A}\|_F, \tag{11}$$

where $\mathbf{A}$ is the adjacency matrix and λ is a constant weight. Intuitively, the second term of Eq. 11 forces $\mathcal{G}$ to minimize the edge weights only at necessary points and keep other edge weights high.

Scalability

A drawback entailed with optimizing $\sigma_{\mathbf{x}}$ as described in the previous sections is that it requires computing the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, which is not scalable to large N. Therefore, we propose an alternative mechanism to compute $\sigma_{\mathbf{x}}$ efficiently. To this end, we make a key observation that optimal $\sigma_{\mathbf{x}}$ approximately depends on $\|\frac{\partial \bar{f}(\mathbf{x})}{\partial \mathbf{x}}\|_2$, where $\bar{f}(\mathbf{x})$ is the mean value of the signal output $f(\mathbf{x})$. To empirically validate this, we visualize the derivative map of the signal against $\sigma_{\mathbf{x}}$. Fig. 3 visualizes the results. As evident, $\sigma_{\mathbf{x}}$ is approximately a point-wise property of the derivative map. Leveraging the above observation, we approximate $\sigma_{\mathbf{x}}$ using the following polynomial equation:

$$\begin{aligned}\sigma_{\mathbf{x}_i} = \beta_1 + \beta_2 \|\frac{\partial \bar{f}(\mathbf{x}_i)}{\partial \mathbf{x}_i}\|_2 \beta_3 \|\frac{\partial \bar{f}(\mathbf{x}_i)}{\partial \mathbf{x}_i}\|_2^2 + \dots \\ + \beta_L \|\frac{\partial \bar{f}(\mathbf{x}_i)}{\partial \mathbf{x}_i}\|_2^{L-1},\end{aligned} \tag{12}$$

The coefficients $\{\beta_i\}_{i=1}^L$ of the Eq. 12 can be found analytically, or via a linear MLP, using a small training set ($\sim$ 20 samples of $\|\frac{\partial \bar{f}(\mathbf{x}_i)}{\partial \mathbf{x}_i}\|_2$ and $\sigma_{\mathbf{x}_i}$ pairs). Once computed, $\{\beta_i\}_{i=1}^L$ can be used universally across similar signal classes, *e.g.*, same coefficients can be used to find $\sigma_{\mathbf{x}}$ for an arbitrary image. We also observe that $L = 10$ is enough to provide adequate results.

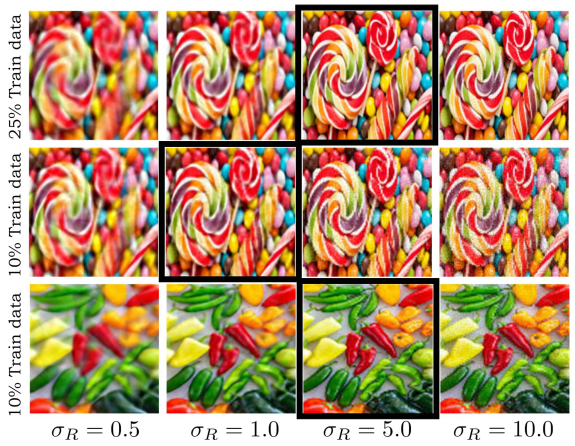

Figure 4: Encoding signals with RFF (zoom-in for a better view). The representations with the highest quality are highlighted with a black border in each row. σ_R denotes the standard deviation of the distribution where the frequency components for RFF are chosen from. As shown, the optimal hyperparameters for RFF vary according to the characteristics of the encoded signal and the sampling procedure, making manual hyperparameter sweeps expensive and ambiguous. In contrast, our method do not suffer from such a drawback.

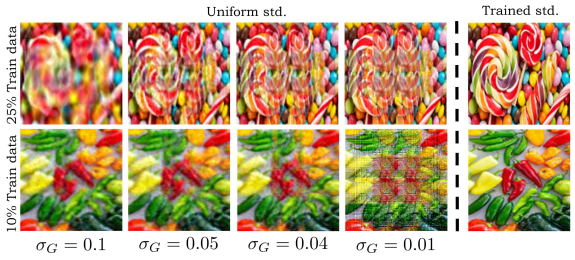

Figure 5: Uniform standard deviations vs trained standard deviations for encoding signals with super-Gaussian embedders (zoom-in for a better view). Note that the different regions of the images show different reconstruction qualities with uniform standard deviations, indicating that they should be optimized per-coordinate. In contrast, the proposed training method achieves better and consistent representation quality across the image.

Experiments

We now conduct experiments to validate the efficacy of the proposed method.

Hyperparameters and Experimental Settings

We first construct three small datasets (20 samples each) of 1D, 2D and 3D signals, and then obtain the pairs

Embedding	Train PSNR	Test PSNR
No PE	20.42	20.39
RFF (matched)	34.53	26.03
RFF (unmatched)	32.16	23.24
super-Gaussian (uni.)	34.52	27.19
super-Gaussian (e.tr.)	39.12	21.44
super-Gaussian (p.tr)	34.59	**31.17**

Table 1: Quantitative comparison in 1D signal encoding. Our method outperforms RFF. The performance of RFF drops when the parameters are not found via cross-validation per signal. (uni.), (e.tr.), and (p.tr.) denotes uniform, end-to-end trained, and proposed method, respectively.

$(\|\frac{\partial \bar{f}(\mathbf{x}_i)}{\partial \mathbf{x}_i}\|_2, \sigma_{\mathbf{x}_i})$ by minimizing the 11. Then, we calculate the coefficient sets $\{\beta\}_{i=1}^{L}$ of Eq. 12 for 1D, 2D, and 3D signals individually using least square optimization. These coefficients are then used throughout all the experiments to find $\sigma_{\mathbf{x}}$ of a given signal using Eq. 12. However, in most cases, it is required to regress to unseen coordinates. In such scenarios, we obtain the approximate first-order gradients for the sparse training points, and calculate the training $\sigma_{\mathbf{x}_{train}}$. Then, $\sigma_{\mathbf{x}_{test}}$ for the test coordinates are determined via linear interpolation. The hyperparameters for the Gaussian embedder and RFF were chosen by manual hyperparameter sweeps, *i.e.* for each signal, we conducted an extensive grid search for optimal hyperparameters. This reconfirms the efficacy of our method. Further, all the MLPs use ReLU activations, and are trained using the Adam optimizer with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-8}.

Encoding Signals

1D signals: First, we pick 25 random rows from the natural 2D images released by (Tancik et al. 2020) to create a small dataset of 1D signals, each with 512 pixels. Then, we encode the coordinates using different embedding layers and feed each of them to a 4-layer MLP to encode the signal. For each signal, we sample 256 points with an interval of one as the training set, and the rest of the points as the testing set. Experiments were repeated 10 times to obtain the average performance for each embedder. Table 1 depicts the results. As shown, our embedder reports the best train, and test PSNRs over RFF when trained using the proposed method. In RFF (matched), we conducted a grid search to obtain the best parameters for each signal. In RFF (unmatched), we find the best parameters for the first signal, and keep it constant across all the signals. As shown, RFF (unmatched) yields inferior performance to the RFF (matched), confirming that the optimal parameters for RFF depend on the signal.

3D signals: For evaluating the embedders on 3D signals, we utilize NeRF-style 3D scenes. The quantitative results are shown in Table 3. RFF (matched) uses log linear sampling of frequencies with the maximum frequency 2^{10}. RFF (unmatched ↑) and RFF (unmatched ↓) use 2^{15} and 2^{8} as the maximum frequency component, respectively. When the chosen frequency components deviate from the ideal values, RFF demonstrates a drop in performance. This behavior confirms the importance of a principled method for finding the optimal parameters for a positional embedding.

Layers	25% tr. data (regular)		
	RFF	super-Gauss (uni.)	super-Gauss (p.tr.)
1	16.89	17.40	**19.99**
2	22.13	20.07	**23.01**
3	23.11	21.13	**23.95**
Layers	10% tr. data (regular)		
	RFF	super-Gauss (uni.)	super-Gauss (p.tr.)
1	15.78	16.17	**18.68**
2	17.76	17.12	**19.51**
3	19.50	18.01	**21.12**
Layers	25% tr. data (random)		
	RFF	super-Gauss (uni.)	super-Gauss (p.tr.)
1	16.01	16.80	**19.33**
2	19.99	18.44	**22.81**
3	21.83	19.14	**23.41**
Layers	10% tr. data (random)		
	RFF	super-Gauss (uni.)	super-Gauss
1	13.17	14.91	**17.64**
2	16.41	15.87	**18.32**
3	18.47	17.11	**20.96**

Table 2: The expressiveness of the embedders tested against different training conditions. The super-Gaussian embedder optimized using the proposed method is able to outperform RFF embedder with both 25% and 10% training data. In particular, trained super-Gaussian embedders perform better with a fewer number of layers. When used with uniformly distributed standard deviations, super-Gaussian embedders demonstrate poor results. All reported values are in mean PSNR. (uni.) and (p.tr.) denotes uniform and proposed method, respectively. we use 120×120 rescaled images from the natural category od (Tancik et al. 2020).

Embedding	PSNR	SSIM
RFF (matched)	31.13	0.947
RFF (unmatched ↑)	29.16	0.911
RFF (unmatched ↓)	25.11	0.801
super-Gaussian (uni.)	25.41	0.891
super-Gaussian (p. tr.)	**32.17**	**0.981**

Table 3: Quantitative comparison embedders in 3D signal encoding on Realistic Synthetic dataset (Mildenhall et al. 2020). Ours yield the best PSNR and SSIM in view synthesis. The performance of RFF drops When the number of unique frequency components differs from the optimal value.

Fig. 4 depicts a qualitative example that further validates the high cost of using cross-validation to obtain optimal parameters for RFF. We test the encoding performance of coordinate-MLPs, when equipped with RFF positional em-

Figure 6: Encoding 3D signals (zoom in for a better view). The left block shows NeRF-style (Mildenhall et al. 2020) examples, where we used the same model as in (Mildenhall et al. 2020) with our embedding. The right block illustrates three frames of an encoded video. When RFF deviates from the optimal parameter setting, the results are considerably inferior to the super-Gaussian embedder. Even with the optimal parameter setting, the super-Gaussian embedder achieves better quantitative results compared to RFF (Table 3).

beddings with varying hyperparameters. As Fig 4 illustrates, the ideal parameters depend on the signal and the sampling scheme, making the cross-validation process difficult and expensive. Therefore, extensive grid searches are required to gain optimal performance. In contrast, our embedder does not suffer from such a drawback. Fig. 5 is a qualitative comparison between uniformly distributed and trained $\sigma_{\mathbf{x}}$. Across all the experiments, we compare the performance variation when using super-Gaussian embedders with uniform, end-to-end trained, and graph-Laplacian trained standard deviations. The end-to-end trained embedder yields high training PSNR, but low test PSNR. Uniform standard deviations provide a better trade-off between the two, but show inferior performance to the proposed training scheme. Across all the experiments, the super-Gaussian embedders trained with the proposed method outperform the other methods consistently.

Expressiveness of the Representation

Recall that a bulk of our derivations stemmed from enforcing the smoothness of the embedding layer with respect to the output. Thus, it is reasonable to hypothesize that our embedder should demonstrate superior performance with shallow networks. To this end, we compare the performance of the embedders while varying the depth of the MLP. We observed consistently better performance with our embedder across different depth settings (Table 2). We also evaluate the robustness of the embedders under different sampling conditions. As illustrated in Table 2, our method outperforms RFF under various sampling schemes. Fig. 2 illustrates qualitative examples.

Stable Gradients

Being able to backpropagate through a module is crucial in deep learning. Stable backpropagation through a module implies that it can be integrated into a deep network as an intermediate layer. As evident from Fig. 7, the proposed embedder is able to produce more stable gradients compared to RFF, especially as the complexity of the signal increases.

Conclusion

We develop a framework that can be used to optimize positional embeddings. We validate the efficacy of our embedder over the popular RFF embedder across various tasks, and show that ours yield better fidelity and stability in different training conditions. Finally, we show that compared to RFF, the super-Gaussian embedder can produce smooth gradients during backpropagation, which allows using embedding layers as intermediate modules in deep networks.

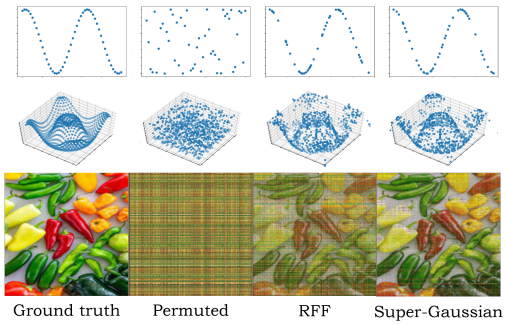

Figure 7: Each row corresponds to different signals with increasing complexity (top to bottom). First, the MLPs are trained to overfit the signal. Then, the coordinates are permuted. Finally, the MLPs try to recover the original coordinates using gradient descent with the original image as the ground truth. As the signal complexity increases, super-Gaussian emebdder recovers the coordinates better compared to RFF.

References

Alliez, P.; Cohen-Steiner, D.; Tong, Y.; and Desbrun, M. 2007. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry processing*, volume 7, 39–48.

Basher, A.; Sarmad, M.; and Boutellier, J. 2021. LightSAL: Lightweight Sign Agnostic Learning for Implicit Surface Representation. *arXiv preprint arXiv:2103.14273*.

Bi, S.; Xu, Z.; Srinivasan, P.; Mildenhall, B.; Sunkavalli, K.; Hašan, M.; Hold-Geoffroy, Y.; Kriegman, D.; and Ramamoorthi, R. 2020. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*.

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

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, Ł. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Deng, B.; Lewis, J. P.; Jeruzalski, T.; Pons-Moll, G.; Hinton, G.; Norouzi, M.; and Tagliasacchi, A. 2020. NASA neural articulated shape approximation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, 612–628. Springer.

Deng, K.; Liu, A.; Zhu, J.-Y.; and Ramanan, D. 2021. Depth-supervised NeRF: Fewer Views and Faster Training for Free. *arXiv preprint arXiv:2107.02791*.

Dupont, E.; Goliński, A.; Alizadeh, M.; Teh, Y. W.; and Doucet, A. 2021. COIN: COmpression with Implicit Neural representations. *arXiv preprint arXiv:2103.03123*.

Genova, K.; Cole, F.; Sud, A.; Sarna, A.; and Funkhouser, T. 2020. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4857–4866.

Guo, M.; Fathi, A.; Wu, J.; and Funkhouser, T. 2020. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*.

Henzler, P.; Mitra, N. J.; and Ritschel, T. 2020. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8356–8364.

Jain, A.; Tancik, M.; and Abbeel, P. 2021. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5885–5894.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942*.

Liu, X.; Yu, H.-F.; Dhillon, I.; and Hsieh, C.-J. 2020. Learning to encode position for transformer with continuous dynamical model. In *International Conference on Machine Learning*, 6327–6335. PMLR.

Martin-Brualla, R.; Radwan, N.; Sajjadi, M. S.; Barron, J. T.; Dosovitskiy, A.; and Duckworth, D. 2021. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7210–7219.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 405–421. Springer.

Mu, J.; Qiu, W.; Kortylewski, A.; Yuille, A.; Vasconcelos, N.; and Wang, X. 2021. A-SDF: Learning Disentangled Signed Distance Functions for Articulated Shape Representation. *arXiv preprint arXiv:2104.07645*.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 427–436.

Niemeyer, M.; Mescheder, L.; Oechsle, M.; and Geiger, A. 2020. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3504–3515.

Oechsle, M.; Mescheder, L.; Niemeyer, M.; Strauss, T.; and Geiger, A. 2019. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4531–4540.

Pang, J.; and Cheung, G. 2017. Graph Laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26(4): 1770–1785.

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

Park, K.; Sinha, U.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Seitz, S. M.; and Martin-Brualla, R. 2021. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5865–5874.

Pumarola, A.; Corona, E.; Pons-Moll, G.; and Moreno-Noguer, F. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10318–10327.

Rahaman, N.; Baratin, A.; Arpit, D.; Draxler, F.; Lin, M.; Hamprecht, F.; Bengio, Y.; and Courville, A. 2019. On the spectral bias of neural networks. In *International Conference on Machine Learning*, 5301–5310. PMLR.

Rahimi, A.; Recht, B.; et al. 2007. Random Features for Large-Scale Kernel Machines. In *NIPS*, volume 3, 5. Citeseer.

Rebain, D.; Jiang, W.; Yazdani, S.; Li, K.; Yi, K. M.; and Tagliasacchi, A. 2021. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14153–14161.

Saito, S.; Huang, Z.; Natsume, R.; Morishima, S.; Kanazawa, A.; and Li, H. 2019. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2304–2314.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; and Wetzstein, G. 2020. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33.

Sitzmann, V.; Zollhöfer, M.; and Wetzstein, G. 2019. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*.

Stanley, K. O. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2): 131–162.

Takikawa, T.; Litalien, J.; Yin, K.; Kreis, K.; Loop, C.; Nowrouzezahrai, D.; Jacobson, A.; McGuire, M.; and Fidler, S. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11358–11367.

Tancik, M.; Srinivasan, P. P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J. T.; and Ng, R. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*.

Tiwari, G.; Sarafianos, N.; Tung, T.; and Pons-Moll, G. 2021. Neural-GIF: Neural Generalized Implicit Functions for Animating People in Clothing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11708–11718.

Trevithick, A.; and Yang, B. 2020. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*.

Wang, B.; Zhao, D.; Lioma, C.; Li, Q.; Zhang, P.; and Simonsen, J. G. 2019. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*.

Wang, Z.; Wu, S.; Xie, W.; Chen, M.; and Prisacariu, V. A. 2021. NeRF–: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064*.

Xiang, F.; Xu, Z.; Hasan, M.; Hold-Geoffroy, Y.; Sunkavalli, K.; and Su, H. 2021. NeuTex: Neural Texture Mapping for Volumetric Neural Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7119–7128.

Xu, H.; Alldieck, T.; and Sminchisescu, C. 2021. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. *Advances in Neural Information Processing Systems*, 34: 14955–14966.

Yen-Chen, L.; Florence, P.; Barron, J. T.; Rodriguez, A.; Isola, P.; and Lin, T.-Y. 2020. inerf: Inverting neural radiance fields for pose estimation. *arXiv preprint arXiv:2012.05877*.

Yu, A.; Li, R.; Tancik, M.; Li, H.; Ng, R.; and Kanazawa, A. 2021a. Plenoctrees for real-time rendering of neural radiance fields. *arXiv preprint arXiv:2103.14024*.

Yu, A.; Ye, V.; Tancik, M.; and Kanazawa, A. 2021b. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4578–4587.

Zheng, J.; Ramasinghe, S.; and Lucey, S. 2021. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*.

Zhong, E. D.; Bepler, T.; Davis, J. H.; and Berger, B. 2019. Reconstructing continuous distributions of 3D protein structure from cryo-EM images. *arXiv preprint arXiv:1909.05215*.

Zhou, D.; and Burges, C. J. 2008. High-order regularization on graphs. In *Proceedings of the 6th International Workshop on Mining and Learning with Graphs*.