Trading Positional Complexity vs Deepness in Coordinate Networks

Jianqiao Zheng ⋆, Sameera Ramasinghe⋆, Xueqian Li, and Simon Lucey

Australian Institute for Machine Learning
University of Adelaide
jianqiao.zheng@adelaide.edu.au

Abstract. It is well noted that coordinate-based MLPs benefit—in terms of preserving high-frequency information—through the encoding of coordinate positions as an array of Fourier features. Hitherto, the rationale for the effectiveness of these *positional encodings* has been mainly studied through a Fourier lens. In this paper, we strive to broaden this understanding by showing that alternative non-Fourier embedding functions can indeed be used for positional encoding. Moreover, we show that their performance is entirely determined by a trade-off between the stable rank of the embedded matrix and the distance preservation between embedded coordinates. We further establish that the now ubiquitous Fourier feature mapping of position is a special case that fulfills these conditions. Consequently, we present a more general theory to analyze positional encoding in terms of shifted basis functions. In addition, we argue that employing a more complex positional encoding—that scales exponentially with the number of modes—requires only a linear (rather than deep) coordinate function to achieve comparable performance. Counter-intuitively, we demonstrate that trading positional embedding complexity for network deepness is orders of magnitude faster than current state-of-the-art; despite the additional embedding complexity. To this end, we develop the necessary theoretical formulae and empirically verify that our theoretical claims hold in practice.
Project page at https://osiriszjq.github.io/complex_encoding.

Keywords: coordinate networks, positional encoding, signal reconstruction

1 Introduction

Positional encoding is an umbrella term used for representing the coordinates of a structured object as a finite-dimensional embedding. Such embeddings are fast becoming critical instruments in modern language models [2, 6, 9, 15, 35, 37] and vision tasks that involve encoding a signal (*e.g.*, 2D image, 3D object, etc.) as weights of a neural network [3, 8, 16, 18, 19, 22, 23, 38]. Of specific interest in this paper is the use of positional encodings when being used to enhance the performance of *coordinate-MLPs*. Coordinate-MLPs are fully connected networks, trained to learn the structure of an object as a continuous function, with coordinates as inputs. However, the major drawback of training coordinate-MLPs with raw input coordinates is their sub-optimal performance in learning high-frequency content [25].

As a remedy, recent studies empirically confirmed that projecting the coordinates to a higher dimensional space using sine and cosine functions of different frequencies (*i.e.*,

⋆ Authors contributed equally.

Fourier frequency mapping) allows coordinate-MLPs to learn high-frequency information more effectively [19,38]. This observation was recently characterized theoretically by Tancik *et al*. [34], showing that the above projection permits tuning the spectrum of the neural tangent kernel (NTK) of the corresponding MLP, thereby enabling the network to learn high-frequency information. Despite impressive empirical results, encoding position through Fourier frequency mapping entails some unenviable attributes. First, prior research substantiates the belief that the performance of the Fourier feature mapping is sensitive to the choice of frequencies. Leading methods for frequency selection, however, employ a stochastic strategy (*i.e*., random sampling) which can become volatile as one attempts to keep to a minimum the number of sampled frequencies. Second, viewing positional encoding solely through a Fourier lens obfuscates some of the fundamental principles behind its effectiveness. These concerns have heightened the need for an extended analysis of positional encoding.

This paper aims to overcome the aforesaid limitations by developing an alternative and more comprehensive understanding of positional encoding. The foremost benefit of our work is allowing non-Fourier embedding functions to be used in the positional encoding. Specifically, we show that positional encoding can be accomplished via systematic sampling of shifted continuous basis functions, where the shifts are determined by the coordinate positions. In comparison to the ambiguous frequency sampling in Fourier feature mapping, we derive a more interpretable relationship between the sampling density and the behavior of the embedding scheme. In particular, we discover that the effectiveness of the proposed embedding scheme primarily relies on two factors: (i) the approximate matrix rank of the embedded representation across positions, and (ii) the distance preservation between the embedded coordinates. Distance preservation measures the extent to which the inner product between the shifted functions correlates with the Euclidean distance between the corresponding coordinates. Intuitively, a higher approximate matrix rank causes better memorization of the training data, while the distance preservation correlates with generalization. Remarkably, we establish that any given continuous function can be used for positional encoding—as performance is simply determined by the trade-off between the aforementioned two factors. Further, we assert that the effectiveness and shortcomings of Fourier feature mapping can also be analyzed in the context of this newly developed framework. We also propose a complex positional encoding to relax the expressibility of the coordinate network into a single linear layer, which largely speedups the instance-based optimization. An essential idea here is the separation of the coordinates. For a simple 1D signal, the input is only embedded in one direction. As for 2D natural images and 3D video sequences, the coordinates are still separable, which enables us to use the Kronecker product to gather input embedding in every single direction. With signals that have non-separable coordinates, we add a blending matrix to linearly interpolate to get the final embedding. In summary, the contribution of this paper is four-fold:

- We expand the current understanding of positional encoding and show that it can be formulated as a systematic sampling scheme of shifted continuous basis functions. Compared to the popular Fourier frequency mapping, our formulation is more interpretative in nature and less restrictive.
- We develop theoretical formulae to show that the performance of the encoding is governed by the approximate rank of the embedding matrix (sampled at different

positions) and the distance preservation between the embedded coordinates. We further solidify this new insight using empirical evaluations.
– As a practical example, we employ a Gaussian signal as the embedding function and show that it can deliver on-par performance with the Fourier frequency mapping. Most importantly, we demonstrate that the Gaussian embedding is more efficient in terms of the embedding dimension while being less volatile.
– We show that trading a complex positional encoding for a deep network allows us to encode high-frequency features with a substantial speedup by circumventing the heavy computation for a simple positional encoding combined with a deep neural network. Promising empirical reconstruction performance is obtained on 1D, 2D, and 3D signals using our proposed embedding function in conjunction with coordinate networks.

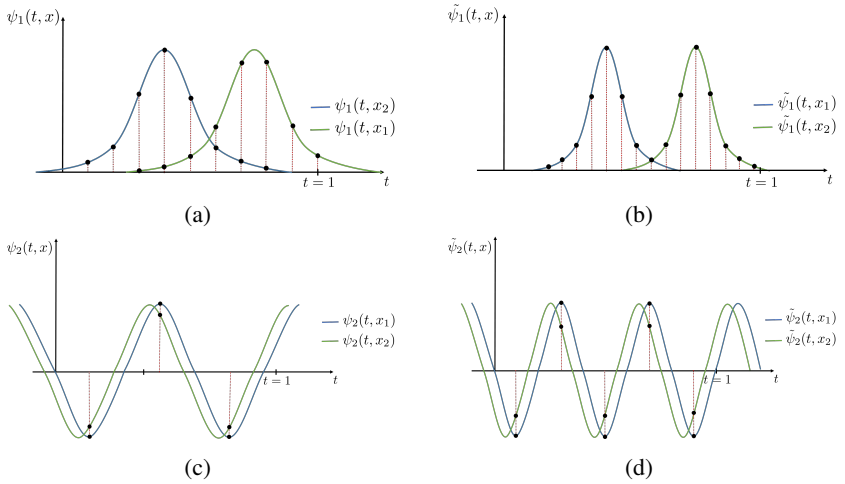


Fig. 1: Overview of the proposed positional encoding scheme. Positions are encoded as equidistant samples from shifted basis functions (embedders). The shifts are determined by the corresponding coordinate positions we are wanting to embed. In (a) and (b), x_1 and x_2 are encoded as samples from shifted Gaussians with a higher and a lower standard deviation, respectively. Note that we need a higher number of samples for (b) due to higher bandwidth (see Section 3). In (c) and (d), x_1 and x_2 are encoded with sinusoidal signals with a different frequencies. Note that although different sampling rates are employed for (c) and (d), the same two values are repeated across the samples. Hence, sampling more than twice is redundant.

2 Related works

Positional encoding became a popular topic among the machine learning community after the seminal work on Transformers by Vaswani *et al*. [35] Since the attention mechanism used in the Transformers is position-insensitive, they employed a sinusoidal signal to encode the positions before feeding them to the higher blocks. A contemporary work by Gehring *et al*. [9] also proposed a convolutional seq2seq model, adapting a positional encoding mechanism. Since then, using positional encoding in language

models became a common trend [5, 10, 13, 24, 30]. Notably, Wang *et al*. [36] extended the embedding space from real numbers to complex values. Another critical aspect of their work is replacing the pre-defined encoding mechanism with a learnable one. There have also been other exciting attempts to improve positional encoding, such as extending the sequential positional encoding to tree-based positional encoding [31], untying the correlations between words and positions while embedding coordinates [12], and modeling positional encoding using dynamical systems [17].

In parallel, positional encoding is also gaining attention in computer vision, specifically with coordinate-MLPs. Coordinate-MLPs provide an efficient method to encode objects such as images [20, 33] and 3D scenes [21, 29, 32] as their weights. Remarkably, Mildenhall *et al*. [19] and Zhong *et al*. [38] found that encoding coordinates with sinusoidal signals allow coordinate-MLPs to learn high frequency content better. One of the earliest roots of this approach can perhaps be traced to the work by Rahimi and Recht [26], where they used random Fourier features to approximate an arbitrary stationary kernel function by applying Bochner's theorem. More recently, Tancik *et al*. [34], leveraging the NTK theory [1, 4, 7, 11, 14], recently added theoretical rigor to this particular practice by showing that such embeddings enable tuning the spectrum of the NTK of the corresponding MLP. In contrast, the goal of this paper is to show that one does not have to be limited to the Fourier embedding for positional encoding. We demonstrate that alternative functions can be used for positional encoding while gaining similar or better performance compared to Fourier embedding.

3 Positional encoding: a theoretical walk-through

This section contains an exposition of the machinery and fundamentals necessary to understand the proposed framework. We begin our analysis by considering a simple linear learner since rigorous characterization of a linear learner is convenient compared to a non-linear model. Therefore, we study a linear learner and empirically show that the gathered insights are extendable to the non-linear models.

First, we show that the capacity to memorize a given set of training data entirely depends on the (approximate) rank of the embedding matrix. Next, we establish that for generalization, the rank should be upper-bounded against the number of coordinates, *i.e*., the embedding function should be bandlimited [1]. We incur a crucial insight here that positional encoding essentially portrays a trade-off between memorization and generalization. Afterward, we discuss the importance of distance preservation between embedded coordinates and its relationship to bandlimited embedding functions. Finally, we consider several possible embedder functions and analyze their behavior using the developed tools.

3.1 Rank of the embedded representation

Let $\mathbf{x}{=}[x_1, x_2, \cdots, x_N]^T$ be a vector of 1D coordinates, in which $x_i{\in}[0, C]$. And let $\mathbf{y}{=}[y_1, y_2, \cdots, y_n]^T$ be the corresponding outputs of a function $f{:}\mathbb{R}{\rightarrow}\mathbb{R}$. Our goal is to find a d dimensional embedding $\Psi{:}\mathbb{R}{\rightarrow}\mathbb{R}^d$ for these positions, so that a linear model can be employed to learn the mapping f as,

[1] We assume that in regression, the smoothness of a model is implicitly related to generalization.

$$\mathbf{w}^T \Psi(\mathbf{x}) + b \approx f(\cdot) \,, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the learnable weights and the bias, respectively. Then, it is straightforward to show that for the perfect reconstruction of *any* given $\mathbf{y}$ using Eq. (1), the following condition should be satisfied:

$$\text{Rank}\left\{[\Psi(x_1)\, \Psi(x_2)\, \ldots\, \Psi(x_N)]\right\} = N \,. \tag{2}$$

Thus, we establish the following Proposition:

Proposition 1 *Consider a set of coordinates* $\mathbf{x}=[x_1, x_2, \cdots, x_N]^T$*, corresponding outputs* $\mathbf{y}=[y_1, y_2, \cdots, y_N]^T$*, and a* d *dimensional embedding* $\Psi:\mathbb{R}\rightarrow\mathbb{R}^d$*. Under perfect convergence, the sufficient condition for a linear model for perfectly memorizing the mapping between* $\mathbf{x}$ *and* $\mathbf{y}$ *is for* $\mathbf{X}=[\Psi(x_1), \Psi(x_2), \ldots, \Psi(x_N)]^T$ *to have full rank.*

3.2 Bandlimited embedders

One possible way of enforcing the condition in Eq. (2) is to define an embedding scheme where the rank of the embedded matrix strictly monotonically increases with N (for a sufficiently large d). As depicted in Section 3.1, this would ensure that the model can memorize the training data and therefore perfectly reconstruct $\mathbf{y}$. However, memorization alone does not yield a good model. On the contrary, we also need our model to be generalizable to unseen coordinates.

To this end, let us define elements of $\Psi(\cdot)$ as sampled values from a function $\psi:\mathbb{R}^2\rightarrow\mathbb{R}$ such that for a given x,

$$\Psi(x) = [\psi(0, x), \psi(s, x), \ldots, \psi((d-1)s, x)]^T \,, \tag{3}$$

where $s=Cd^{-1}$ is the sampling interval. We shall refer to $\psi(\cdot)$ as the *embedder*. As discussed above, for better generalization, we need,

$$\psi(t, x) \approx \sum_{b=0}^{B} \alpha_b \beta_b(t) \,, \tag{4}$$

where α_b and $\beta_b(t)$ are weights and shifted basis functions, respectively, that can approximately estimate $\psi(t, x)$ at any arbitrary position x. We refer to such embedders as *bandlimited embedders* with a bandwidth B. This is equivalent to saying that the embedding matrix has a bounded rank, *i.e.*, the rank cannot increase arbitrarily with N. The intuition here is that if B is too small, the model will demonstrate poor memorization and overly smooth generalization. On the other hand, if B is extremely high, the model is capable of perfect memorization but poor generalization. Therefore we conclude that for ideal performance, the embedder should be chosen carefully, such that it is both bandlimited and has a sufficient rank. As we shall discuss the bandwidth B can also act as a guide for the minimal value of d.

3.3 Distance preservation

Intuitively, the embedded coordinates should preserve the distance between the original coordinates, irrespective of the absolute position. The embedded distance (or similarity) $\mathrm{D}(\cdot, \cdot)$ between two coordinates (x_1, x_2) can be measured via the inner product $\mathrm{D}(x_1, x_2) = \int_0^1 \psi(t, x_1)\psi(t, x_2)dt$. For ideal distance preservation we need,

$$\|x_1 - x_2\| \propto \mathrm{D}(x_1, x_2)\ . \tag{5}$$

Interestingly, this property is also implicitly related to the limited bandwidth requirement. Note that in practice, we employ sampled embedders to construct Ψ as shown in Eq. (3). Hence, the dot product between the sampled $\psi(t, x_1)$ and $\psi(t, x_2)$ should be able to approximate D as,

$$\mathrm{D}(x_1, x_2) = \int_0^C \psi(t, x_1)\psi(t, x_2)dt \approx \sum_{d=0}^{d-1} \psi(s \cdot d, x_1)\psi(s \cdot d, x_2)\ , \tag{6}$$

which is possible, if and only if, ψ is bandlimited. In that case, $d{=}B$ is sufficient where B is the bandwidth of ψ (by Nyquist sampling theory). In practice, we choose $C{=}1$.

Remark 1. The embedder should be bandlimited for better generalization (equivalently, the rank of the embedded matrix should be upper-bounded). Further, the ideal embedder should essentially face a trade-off between memorization and generalization. Here, memorization correlates with the rank of the embedded matrix, while generalization relates to the distance preservation between the embedded coordinates.

4 Analysis of possible embedders

Although our derivations in Section 3 are generic, it is imperative to carefully choose a specific form of $\psi(\cdot, \cdot)$, such that properties of candidate embedders can be conveniently analyzed. Hence, we define embedders in terms of shifted basis functions, *i.e.*, $\psi(t, x){=}\psi(t{-}x)$. Such a definition permits us to examine embedders in a unified manner, as we shall see below.

Moreover, the rank of a matrix can be extremely noisy in practice. Typically, we need to heuristically set an appropriate threshold to the singular values, leading to unstable calculations. Therefore, we use the stable rank [28] instead of the rank in all our experiments. In particular, the stable rank is a more stable surrogate for the rank, and is defined as $\frac{\|\mathbf{A}\|_F^2}{\|\mathbf{A}\|_2^2}$, where $\mathbf{A}$ is the matrix, $\|\cdot\|_F$ is the Frobenius norm, and $\|\cdot\|_2$ is the matrix norm. From here onwards, we will use the terms rank, approximate rank, and stable rank interchangeably.

Impulse embedder. One simple way to satisfy the condition of Eq. (2) for an arbitrary large N is to define $\psi(t, x){=}\delta(t{-}x)$, where $\delta(\cdot)$ is the impulse function. Note that using an impulse embedder essentially converts the embedding matrix to a set of one-hot encodings. With the impulse embedder, we can perfectly memorize a given set of data points, as the embedded matrix has full rank. The obvious drawback, however, is that the bandwidth of the impulse embedder is infinite, *i.e.*, assuming a continuous domain, d needs to reach infinity to learn outputs for all possible positions. Hence, the distance preservation is hampered, and consequently, the learned model lacks generalization.

Rectangle embedder. As an approximation of impulse function (unit pulse), rectangular function $rect(x){=}1$ when $|x|{<}\frac{1}{2}$ and $rect(x){=}0$ when $|x|{>}\frac{1}{2}$. We can define $\psi(x){=}rect\left(\frac{x-t}{d}\right)$, where d is the width of the impulse. Immediately we know the stable rank of rectangle embedder is $\min\left(N, \frac{1}{d}\right)$, where N is the number of sampled coordinates, the distance function $D(x_1, x_2){=}tri(x_1{-}x_2)$, where $tri(\cdot)$ is triangular function. A physical way to understand rectangle embedder is nearest neighbour regression.

Triangle embedder. A better choice to approximate impulse function may be triangular function, which is defined as $tri(x)=\max(1-|x|,0)$. Thus the embedder is defined as $\psi(x)=tri\left(\frac{x-t}{0.5d}\right)$. Here the factor 0.5 makes the width of the triangle to be d. The stable rank of triangular embedder is $\min\left(N,\frac{4/3}{d}\right)$. When d is the same, triangle embedder has a higher stable rank than rectangle embedder. The distance function of triangular embedder is $D(x_1,x_2)=\frac{1}{4}\max\left(d-|x_1-x_2|,0\right)^2$. This distance function looks really like Gaussian function, as illustrated in Fig. 3. A physical way to understand triangle embedder is linear interpolation.

Sine embedder. Consider $\psi(t,x)=\sin(f(t-x))$ for an arbitrary fixed function f. Since $\sin(f(t-x))=\sin(ft)\cos(fx)-\cos(ft)\sin(fx)$, elements of any row of the embedding matrix can be written as a linear combination of the corresponding $\sin(ft)$ and $\cos(ft)$. Thus, the rank of the embedding matrix is upper-bounded at 2. Consequently, the expressiveness of the encoding is limited, leading to poor memorization and overly smooth generalization (interpolation) at unseen coordinates.

Square embedder. Let us denote a square wave with unit amplitude and period 2π as $\mathrm{sgn}(\sin(t))$, where sgn is the sign function. Then, define $\psi(t,x)=\mathrm{sgn}(\sin(t-x))$. It is easy to deduce that the embedded distance $D(x_1,x_2)=1-2\|x_1-x_2\|, \forall|x|\leq 1$ which implies perfect distance preservation. The drawback, however, is that the square wave is not bandlimited. Thus, it cannot approximate the inner product $\int\psi(t,x)\psi(t,x')$ using a finite set of samples as in Eq. (6). However, an interesting attribute of the square wave is that it can be decomposed into a series of sine waves with odd-integer harmonic frequencies as $\mathrm{sgn}(\sin(t))=\frac{4}{\pi}\left[\sin(t)+\frac{1}{3}\sin(3t)+\frac{1}{5}\sin(5t)+\frac{1}{7}\sin(7t)+\dots\right]$. In other words, its highest energy (from a signal processing perspective) is contained in a sinusoidal with the same frequency. Thus, the square wave can be *almost* approximated by a sinusoidal signal. In fact, the square wave and the sinusoidal demonstrate similar properties in terms of the stable rank and the distance preservation (see Fig. 3).

Gaussian embedder. We define the Gaussian embedder as $\psi(t,x)=\exp\left(-\frac{\|t-x\|^2}{2\sigma^2}\right)$ where σ is the standard deviation. The Gaussian embedder is also approximately bandlimited like the square embedder. However, the Gaussian embedder has a higher upper bound for the stable rank that can be controlled by σ. More precisely, when the embedding dimension is large enough, the stable rank of the Gaussian embedding matrix and the embedded distance between coordinates can be obtained analytically as shown below.

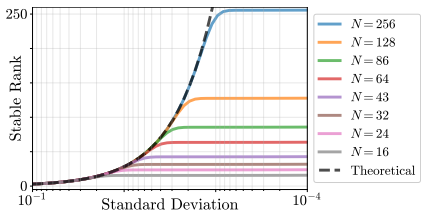
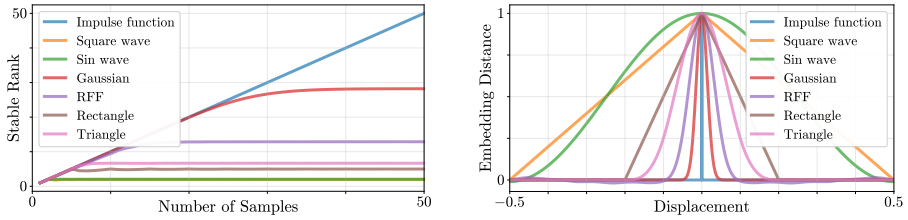


Fig. 2: Stable rank of the Gaussian embedder vs the standard deviation for different number of samples. The dash line is the theoretical stable rank $\frac{1}{2\sqrt{\pi}\sigma}$

Proposition 2 *Let the Gaussian embedder be denoted as* $\psi(t,x)=\exp\left(-\frac{\|t-x\|^2}{2\sigma^2}\right)$. *With a sufficient embedding dimension, the stable rank of the embedding matrix obtained using the Gaussian embedder is* $\min\left(N,\frac{1}{2\sqrt{\pi}\sigma}\right)$ *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1,x_2)=\exp\left(-\frac{\|x_1-x_2\|^2}{4\sigma^2}\right)$.



(a) (b)

Fig. 3: Quantitative comparison of (a) the stable rank and (b) the distance preservation of different embedders and RFFs. As expected, the stable rank of the impulse embedder strictly increases with the number of sampled points, causing poor distance preservation. The stable rank of the sine embedder is upper-bounded at 2. The stable ranks of the square embedder and the sine embedder almost overlap. However, if the sample numbers are extremely high (not shown in the figure), their stable ranks begin to deviate. Similarly, the square embedder demonstrates perfect distance preservation, and the sine embedder is a close competitor. In contrast, the Gaussian embedder and the RFF showcase mid-range upper bounds for the stable rank and adequate distance preservation, advocating a much better trade-off between memorization and generalization.

(see Fig. 2 for an experimental illustration). It is clear from Proposition 2 that as the number of sampled positions goes up, the stable rank of the Gaussian embedding matrix will linearly increase until it reaches its upper bound. Finally, Fig. 3 empirically validates the theoretically discussed properties of different embedders.

4.1 Connection to the Random Fourier Features

The prominent way of employing Fourier frequency mapping is via Random Fourier Features (RFF) mapping [34], where the frequencies are randomly sampled from a Gaussian distribution with a certain standard deviation σ. In this Section, we show that RFF mapping can be analyzed through the lens of our theoretical framework discussed thus far. To this end, we first establish the following proposition:

Proposition 3 *Let the RFF embedding be denoted as* $\gamma(x)=[\cos(2\pi \mathbf{b}x), \sin(2\pi \mathbf{b}x)]$, *where* $\mathbf{b}$ *are sampled from a Gaussian distribution. When the embedding dimension is large enough, the stable rank of RFF will be* $\min\left(N, \sqrt{2\pi}\sigma\right)$, *where* N *is the numnber of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1, x_2)=\sum_j \cos 2\pi b_j(x_1-x_2)$.

As shown in Fig. 4, the stable rank of RFF increases linearly with the number of samples until it gets saturated at $\sqrt{2\pi}\sigma$. This indicates a relationship between RFF and Gaussian embedder. Let σ_g and σ_f be the standard deviations of Gaussian embedder and RFF. When their stable ranks are equal, $\frac{1}{2\sqrt{\pi}\sigma_g}=\sqrt{2\pi}\sigma_f$ (from Proposition 2, 3). This implies that when $\sigma_g\sigma_f=\frac{1}{2\sqrt{2\pi}}$, these two embedders are equivalent in terms of the stable rank and distance preservation (see Fig. 4 when $\sigma_g=0.01$ and $\sigma_f=0.1$).

A common observation with RFFs is that when σ_f is too low, the reconstruction is overly smooth and if σ_f is too high, it gives noisy interpolation [34]. This observation

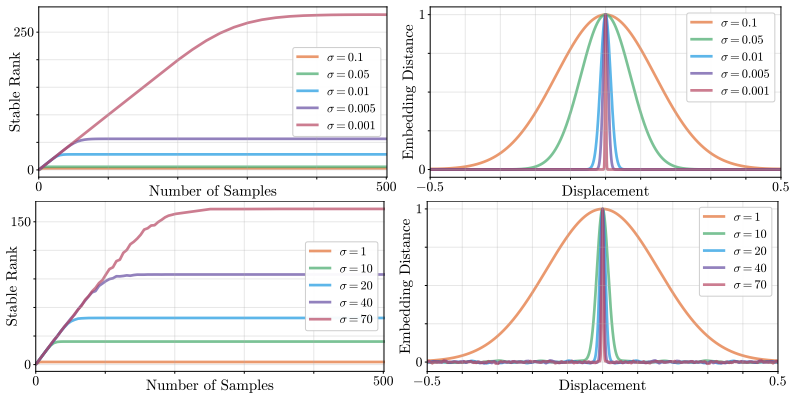


Fig. 4: The stable rank (left column) and distance preservation (right column) of Gaussian embedder (top row) and RFF (bottom row) across different standard deviations.

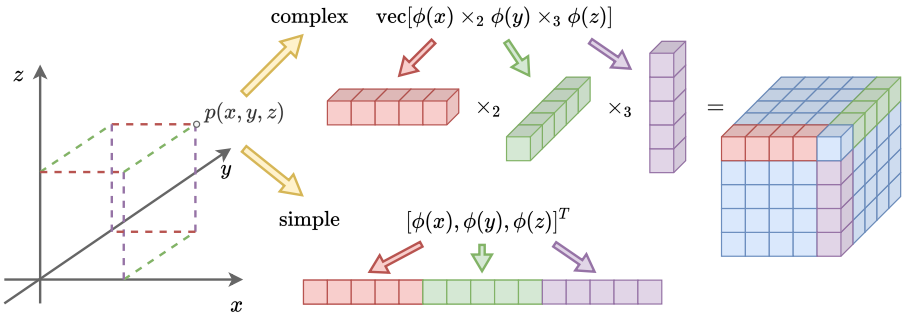


Fig. 5: Illustration of different methods to extend 1D encoding of length K to N-dimensional case. Simple encoding is the widely used method currently, which is only the concatenation of embeddings in each dimension to get a NK size encoding. We propose an alternative embedding method: conduct mode-n product to get the N dimensional cube, which we define as the complex embedding.

directly correlates to our theory. In Fig. 4, as the standard deviation increases, the stable rank increases and distance preservation decreases. Similarly, When the standard deviation is too low, the stable rank decreases while distance preservation increases.

5 Simplicity vs complexity in positional encoding

Thus far, we established that the positional encoding can achieved by sampling shifted basis functions, and the well-known RFF-embedding can also be analyzed through this lens. However, the analysis so far focused only on 1D coordinates. In this section, we shall investigate how to extend these positional embedding schemes to high-dimensional signals, *e.g.*, images and 3D signals.

5.1 2D simple encoding

Suppose $\psi(\cdot)$ is a an arbitrary positional encoding function. We define simple positional encoding as the concatenation of the 1D encoding in each dimension: $\Psi(x, y) = [\psi(x), \psi(y)]$. Then, with a linear model we have,

$$I(x, y) \approx \mathbf{w}^T \Psi(x, y) = \mathbf{w}_x^T \psi(x) + \mathbf{w}_y^T \psi(y) \,. \tag{7}$$

The above formula can be written in the matrix form as,

$$\begin{bmatrix} I(1,1) & \dots & I(N,1) \\ \vdots & \ddots & \vdots \\ I(1,N) & \dots & I(N,N) \end{bmatrix} \approx \underbrace{\begin{bmatrix} \mathbf{w}_x^T \psi(x_1) & \dots & \mathbf{w}_x^T \psi(x_N) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_x^T \psi(x_1) & \dots & \mathbf{w}_x^T \psi(x_N) \end{bmatrix}}_{\mathbf{A}} + \underbrace{\begin{bmatrix} \mathbf{w}_y^T \psi(y_1) & \dots & \mathbf{w}_y^T \psi(y_1) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_y^T \psi(y_N) & \dots & \mathbf{w}_y^T \psi(y_N) \end{bmatrix}}_{\mathbf{B}} \,. \tag{8}$$

Clearly, $\mathbf{A}$ and $\mathbf{B}$ are rank 1 matrices. Therefore, a linear network can only reconstruct a 2D image signal with at most rank 2. This drawback can be addressed in most practical cases using deeper non-linear MLPs, since the rank of the representations can be increased with multiple layers.

5.2 2D complex encoding

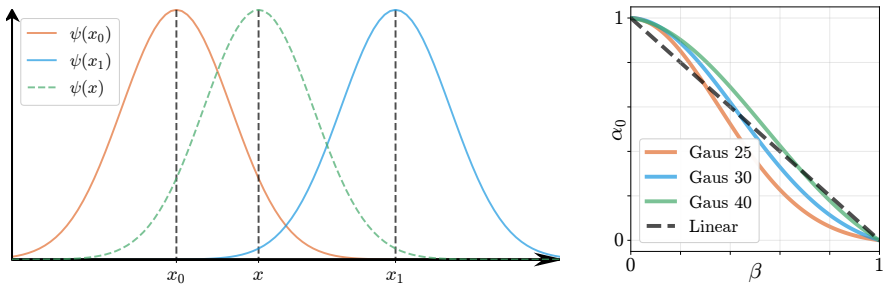
As opposed to simple encoding, we propose an alternative method for positional encoding in higher dimensions using the Kronecker product. With this approach, we can obtain a higher rank for the positional embedding matrix. For example, consider 2D inputs. Then, we can obtain the complex encoding as $\Psi(\mathbf{x}, \mathbf{y}){=}\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y})$, where $\Psi(\mathbf{x})$ and $\Psi(\mathbf{y})$ are 1D encodings along each dimension. Also, the following relationship holds:

$$\mathrm{Rank}(\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y})) = \mathrm{Rank}(\Psi(\mathbf{x}))\mathrm{Rank}(\Psi(\mathbf{y})) \,. \tag{9}$$

However, the drawback is also obvious. The embedding dimension is squared, which takes significantly more memory and computational cost. However, we propose an elegant workaround for this problem given that the points are sampled on a regular grid, *i.e.*, when the coordinates are separable, using the following property of the Kronecker product,

$$\mathrm{vec}(\mathrm{S})^T \approx \mathrm{vec}(\mathbf{W})^T (\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y})) = \mathrm{vec}(\Psi(\mathbf{y})^T \mathbf{W} \Psi(\mathbf{x}))^T \,, \tag{10}$$

where $\mathrm{S}_{i,j}{=}I(x_i, y_j)$. For instance, suppose we have N^2 number of 2D separable points where the feature length is K for each dimension. The naive Kronecker product leads to $O(N^2K^2)$ computational complexity and $O(N^2K^2)$ memory complexity. Using Eq. (10), we reduce it dramatically to $O(NK(N + K))$ computational complexity and $O(2NK + K^2)$ memory complexity. The key advantage of the complex encoding mechanism is that although it leads to a larger encoding matrix, the ability to achieve full rank allows us to use a single linear layer instead of an MLP, which reduces the computational cost of the neural network substantially. In addition, this enables us to obtain a closed-form solution instead of using stochastic gradient descent, leading to dramatically faster optimization. More precisely, we need to solve,



(a) (b)

Fig. 6: **Relationship between α_0 and β.** (a) is an example positional encoding at x_0, x_1 and x. x-axis is along the feature dimension and y-axis is the encoding value. Here, we want to express $\psi(x)$ as a linear combination of $\psi(x_0)$ and $\psi(x_1)$. (b) is the relationship between the interpolation weight α_0 and the position ratio β. Instead of simple linear interpolation (dash line), the actual relation depends on the encoding function.

$$\arg\min_{\mathbf{W}} \|\mathrm{vec}(\mathrm{S}) - (\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y}))^T \mathrm{vec}(\mathbf{W})\|_2^2 \,, \tag{11}$$

where the solution can be obtained analytically as,

$$\mathbf{W} = (\Psi(\mathbf{y})\Psi(\mathbf{y})^T)^{-1}\Psi(\mathbf{y})\mathrm{S}\Psi(\mathbf{x})^T(\Psi(\mathbf{x})\Psi(\mathbf{x})^T)^{-1} \,. \tag{12}$$

When the coordinates are not separable, we can still take advantage of Eq. (10) by adding a sparse blending matrix B as,

$$\mathrm{vec}(I(\mathbf{x},\mathbf{y}))^T \approx \mathrm{vec}(\mathbf{W})^T(\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y}))B = \mathrm{vec}(\Psi(\mathbf{y})^T\mathbf{W}\Psi(\mathbf{x}))^T B \,. \tag{13}$$

This procedure is equivalent to evaluating virtual points (sampled on a regular grid) and interpolating to arbitrary coordinates using interpolation weights of B. The nature of the interpolation depends on the basis function used for positional encoding. Suppose x_0 and x_1 are two grid points and $x_0 \leq x \leq x_1$. We want $\psi(x) \approx \alpha_0\psi(x_0) + \alpha_1\psi(x_1)$. It can be solved by

$$\arg\min_{\alpha} \|\psi(x) - \begin{bmatrix}\psi(x_0) & \psi(x_1)\end{bmatrix} \alpha\|_2^2 \,, \tag{14}$$

where $\alpha = \begin{bmatrix}\alpha_0 & \alpha_1\end{bmatrix}^T$. With the definition $D(x)$ in Section 3.1, consider $d = x_1 - x_0$, which is the grid interval. Then, $x = x_0 + \beta d$, where $0 \leq \beta \leq 1$, and we have

$$\alpha = \frac{1}{D^2(0) - D^2(d)} \begin{bmatrix} D(0) & -D(d) \\ -D(d) & D(0) \end{bmatrix} \begin{bmatrix} D(\beta d) \\ D((1-\beta)d) \end{bmatrix} \,. \tag{15}$$

Please refer to the supplementary material for more details.

5.3 High dimensional encoding

Both simple and complex encoding methods discussed in the previous sections can be extended easily to arbitrarily high dimensions. Suppose $\mathbf{x} = [x_1, x_2, \cdots, x_D]^T$ are the coordinates of a D dimensional point, $\psi: \mathbb{R} \to \mathbb{R}^K$ is the 1D encoder and $\Psi^{(D)}(\cdot)$ is the D dimensional encoding function. Then, the simple encoding is

Table 1: Performance of natural 2D image reconstruction without random sampling (separable coordinates). ● are simple positional encodings. ● are complex positional encodings with gradient descent. ● are complex positional encodings with closed-form solution. We did not show the result for the complex encoding LinF, LogF and RFF, since they are ill-conditioned (rank deficient) and unstable.

	PSNR	No. of params (memory)	Time (s)
● LinF	26.12 ± 3.99	$329,475$ $(1.32M)$	22.59
● LogF	26.11 ± 4.04	$329,475$ $(1.32M)$	22.67
● RFF [34]	$\mathbf{26.58 \pm 4.18}$	$329,475$ $(1.32M)$	22.62
● Tri	25.96 ± 3.88	$329,475$ $(1.32M)$	22.98
● Gau	26.17 ± 4.17	$329,475$ $(1.32M)$	23.12
● LinF	19.73 ± 2.49	$196,608$ $(0.79M)$	1.93
● LogF	23.55 ± 2.99	$196,608$ $(0.79M)$	1.85
● RFF [34]	24.34 ± 3.24	$196,608$ $(0.79M)$	1.55
● Tri	26.65 ± 3.57	$196,608$ $(0.79M)$	1.48
● Gau	$\mathbf{26.69 \pm 3.74}$	$196,608$ $(0.79M)$	2.01
● Tri	26.36 ± 3.39	$196,608$ $(0.79M)$	0.03
● Gau	$\mathbf{26.69 \pm 3.76}$	$196,608$ $(0.79M)$	0.18

$$\Psi^{(D)}(\mathbf{x}) = [\psi(x_1), \psi(x_2), \cdots, \psi(x_D)]^T . \tag{16}$$

Similarly, the complex encoding can be obtained via the Kronecker product between each encoding as,

$$\Psi^{(D)}(\mathbf{x}) = \text{vec}([\psi(x_1) \times_2 \psi(x_2) \times_3 \cdots \psi(x_3) \cdots \times_D \psi(x_D)]) . \tag{17}$$

Then, we can again extend the workaround we used in Eq. 10 to multiple dimensions as,

$$\text{vec}(\mathbf{W})^T \Psi^{(D)}(\mathbf{x}) = \mathbf{W} \times_1 \psi(x_1) \times_2 \psi(x_2) \cdots \times_D \psi(x_D) . \tag{18}$$

Fig. 5 graphically illustrates the simple and complex positional encoding schemes.

6 Experiments

In this Section, we empirically confirm the advantages of using the proposed embedding procedure and verify that the theoretically predicted properties in the previous sections hold in practice. To this end, five encoding methods are compared: linearly sampled frequency (LinF), log-linearly sampled frequency (LogF), RFF, shifted Gaussian (Gau), and shifted triangle encoder (Tri).

6.1 1D: Rank of input & depth of Network

In this experiment, we randomly sample 16 columns and 16 rows from 512×512 natural images from the image dataset in [34]. And we used 256 equally spaced points for training and the rest 256 points for testing. The encoding length is set to be 256, the same as the number of training points. Parameters were chosen carefully for each encoder to show the relationship of the encoding rank and the depth of the network. The depth of the network changes from 0 (linear layer) to 2 with a fixed width of 256. From Fig. 4, we already know that the rank of the encoding matrix drops when σ of Gau increases or σ of RFF decreases. The result in Fig. 7 shows that when

Table 2: Performance of 2D image reconstruction with randomly sampled inputs (non-separable coordinates). ● are simple positional encodings. ● are complex positional encodings with stochastic gradient descent using smart indexing. The time of LinF and LogF is longer since we don't have the closed-form blending matrix.

	PSNR	No. of params (memory)	Time (s)
● LinF	24.58 ± 3.74	$329,475$ $(1.32M)$	21.97
● LogF	24.76 ± 3.82	$329,475$ $(1.32M)$	22.13
● RFF [34]	$\mathbf{24.90 \pm 3.78}$	$329,475$ $(1.32M)$	22.28
● Tri	24.24 ± 3.56	$329,475$ $(1.32M)$	22.50
● Gau	24.63 ± 3.70	$329,475$ $(1.32M)$	22.86
● LinF	19.64 ± 2.04	$196,608$ $(0.79M)$	11.51
● LogF	22.35 ± 2.82	$196,608$ $(0.79M)$	11.52
● RFF [34]	20.54 ± 2.61	$196,608$ $(0.79M)$	3.08
● Tri	23.21 ± 3.58	$196,608$ $(0.79M)$	1.90
● Gau	$\mathbf{23.97 \pm 3.44}$	$196,608$ $(0.79M)$	2.51

the rank is high, a linear network (0 depth) also works well. When the rank drops (*e.g.*, when the σ of Gau changes from 0.003 to 0.01 to 0.07), the performance of a single linear layer also drops. Adding one layer to the linear network makes up for the performance drop while adding more layers does not help a lot. In conclusion, as the rank drops, a deeper network is needed for better performance.

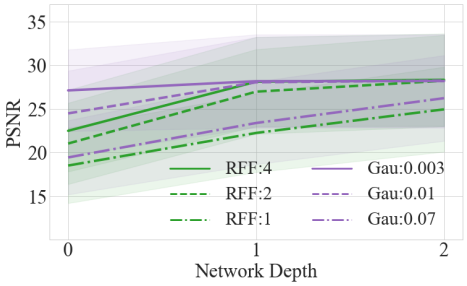


Fig. 7: The performance of the reconstruction depends on the deepness of the network and the rank of the input embedding.

6.2 2D: image reconstruction

For this experiment, we used the image dataset in [34], which contains 32 natural images of size 512×512. For simple encoding, we used a 4 layer MLP with hidden units of 256 widths, and for complex encoding, we only used a single linear layer. The results discussed below are the average metrics for 16 images where the networks are trained for 2000 epochs.

Separable coordinates. 256×256 grid points were evenly sampled for training and the rest were used for testing. As shown in Table 1, complex encoding is around 20 times faster compared to simple encoding. In fact, although both the encodings were trained for 2000 epochs, we observed that complex encodings achieved good performance in a significantly lower number of epochs. Complex encoding can also be solved in closed-form without training, which is orders of magnitude faster than simple encoding and maintains a good performance. Frequency encodings (LinF, LogF, RFF) did not perform well with complex encoding since they were rank deficient. Although complex frequency encodings did not perform as well as complex shifted encoding when combined with a single linear layer, they still outperformed simple encodings followed by a single linear layer.

Non-separable coordinates. For non-separable coordinates, the training points were randomly sampled 25% of the 512×512 natural images. Complex encoding contains a

Table 3: Performance of video reconstruction without random sampling (separable coordinates). ● are simple positional encodings. ● are complex positional encodings with gradient descent; ● are complex positional encodings with closed-form solution.

	PSNR	No. of params (memory)	Time (s)
● LinF	$\mathbf{21.49 \pm 3.29}$	$1,445,891$ $(5.78M)$	77.06
● LogF	21.46 ± 3.09	$1,445,891$ $(5.78M)$	77.86
● RFF [34]	21.06 ± 3.26	$1,445,891$ $(5.78M)$	77.86
● Tri	20.86 ± 3.09	$1,445,891$ $(5.78M)$	78.91
● Gau	21.21 ± 3.26	$1,445,891$ $(5.78M)$	79.09
● LinF	9.76 ± 3.95	$786,432$ $(3.15M)$	0.69
● LogF	9.98 ± 3.84	$786,432$ $(3.15M)$	0.68
● RFF [34]	20.33 ± 3.06	$786,432$ $(3.15M)$	0.69
● Tri	$\mathbf{22.24 \pm 2.89}$	$786,432$ $(3.15M)$	0.68
● Gau	22.11 ± 3.14	$786,432$ $(3.15M)$	0.66
● RFF [34]	18.05 ± 2.36	$786,432$ $(3.15M)$	0.01
● Tri	$\mathbf{22.04 \pm 2.95}$	$786,432$ $(3.15M)$	0.009
● Gau	21.83 ± 3.33	$786,432$ $(3.15M)$	0.008

blending matrix representing virtual separable coordinates, which can be pre-computed. As illustrated in Table 2, the performance of complex encodings was comparably well and converged faster than the simple encodings.

6.3 3D: video reconstruction

As dimensionality of data increases, the faster convergence of the complex encoding becomes more notable. We use the Youtube video [27] for our experiments. In our experiments, we extracted 256 frames from 5 different videos and rescaled each frame to 256×256. Then, a central $128\times128\times128$ cube was cropped to create our dataset, since some videos contain borders. For simple encoding, a 5 layer MLP with 512-width was used, while for complex encoding, only a single linear layer was used. We trained the networks for 500 epochs. The training points were regularly sampled from a $64\times64\times64$ grid and the rest of the points were used for testing. The results are shown in Table 3.

7 Conclusion

In this paper, we show that the performance of a positional encoding scheme is mainly governed by the stable rank of the embedding matrix and the distance preservation between the embedded coordinates. In light of this discovery, we propose a novel framework that can incorporate arbitrary continuous signals as potential embedders, under certain constraints. We also propose a positional encoding scheme that enables dramatically faster convergence, allowing a single linear network to encode signals with high fidelity.

References

1. Arora, S., Du, S., Hu, W., Li, Z., Wang, R.: Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In: International Conference on Machine Learning. pp. 322–332. PMLR (2019) 4
2. Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Gao, J., Piao, S., Zhou, M., et al.: Unilmv2: Pseudo-masked language models for unified language model pre-training. In: International Conference on Machine Learning. pp. 642–652. PMLR (2020) 1
3. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. arXiv preprint arXiv:2103.13415 (2021) 1
4. Bietti, A., Mairal, J.: On the inductive bias of neural tangent kernels. arXiv preprint arXiv:1905.12173 (2019) 4
5. Dai, Z., Yang, Z., Yang, Y., Cohen, W., Carbonell, J., Le, Q., Salakhutdinov, R.T.X.: Attentive language models beyond a fixed-length context. arxiv 2019. arXiv preprint arXiv:1901.02860 4
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018) 1
7. Du, S.S., Zhai, X., Poczos, B., Singh, A.: Gradient descent provably optimizes over-parameterized neural networks. arXiv preprint arXiv:1810.02054 (2018) 4
8. Gafni, G., Thies, J., Zollhöfer, M., Nießner, M.: Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. arXiv preprint arXiv:2012.03065 (2020) 1
9. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: International Conference on Machine Learning. pp. 1243–1252. PMLR (2017) 1, 3
10. He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention. arXiv preprint arXiv:2006.03654 (2020) 4
11. Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. arXiv preprint arXiv:1806.07572 (2018) 4
12. Ke, G., He, D., Liu, T.Y.: Rethinking positional encoding in language pre-training. arXiv preprint arXiv:2006.15595 (2020) 4
13. Kitaev, N., Klein, D.: Constituency parsing with a self-attentive encoder. arXiv preprint arXiv:1805.01052 (2018) 4
14. Lee, J., Xiao, L., Schoenholz, S.S., Bahri, Y., Novak, R., Sohl-Dickstein, J., Pennington, J.: Wide neural networks of any depth evolve as linear models under gradient descent. arXiv preprint arXiv:1902.06720 (2019) 4
15. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019) 1
16. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. arXiv preprint arXiv:2011.13084 (2020) 1
17. Liu, X., Yu, H.F., Dhillon, I., Hsieh, C.J.: Learning to encode position for transformer with continuous dynamical model. In: International Conference on Machine Learning. pp. 6327–6335. PMLR (2020) 4
18. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. arXiv preprint arXiv:2008.02268 (2020) 1
19. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision. pp. 405–421. Springer (2020) 1, 2, 4

20. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 427–436 (2015) 4
21. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3504–3515 (2020) 4
22. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. arXiv preprint arXiv:2011.10379 (2020) 1
23. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Brualla, R.M.: Deformable neural radiance fields. arXiv preprint arXiv:2011.12948 (2020) 1
24. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019) 4
25. Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., Courville, A.: On the spectral bias of neural networks. In: International Conference on Machine Learning. pp. 5301–5310. PMLR (2019) 1
26. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. Advances in neural information processing systems **20** (2007) 4
27. Real, E., Shlens, J., Mazzocchi, S., Pan, X., Vanhoucke, V.: Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5296–5305 (2017) 14
28. Rudelson, M., Vershynin, R.: Sampling from large matrices: An approach through geometric functional analysis. Journal of the ACM (JACM) **54**(4), 12–es (2007) 6
29. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2304–2314 (2019) 4
30. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. arXiv preprint arXiv:1803.02155 (2018) 4
31. Shiv, V., Quirk, C.: Novel positional encodings to enable tree-based transformers. Advances in Neural Information Processing Systems **32**, 12081–12091 (2019) 4
32. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. arXiv preprint arXiv:1906.01618 (2019) 4
33. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic programming and evolvable machines **8**(2), 131–162 (2007) 4
34. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739 (2020) 2, 4, 8, 12, 13, 14
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv preprint arXiv:1706.03762 (2017) 1, 3
36. Wang, B., Zhao, D., Lioma, C., Li, Q., Zhang, P., Simonsen, J.G.: Encoding word order in complex embeddings. arXiv preprint arXiv:1912.12333 (2019) 4
37. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237 (2019) 1
38. Zhong, E.D., Bepler, T., Davis, J.H., Berger, B.: Reconstructing continuous distributions of 3d protein structure from cryo-em images. arXiv preprint arXiv:1909.05215 (2019) 1, 2, 4