RETHINKING POSITIONAL ENCODING

Anonymous authors
Paper under double-blind review

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. To this end, we develop the necessary theoretical formulae and empirically verify that our theoretical claims hold in practice.

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 [1; 2; 3; 4; 5; 6] and vision tasks that involve encoding a signal (e.g. 2D image, 3D object, etc.) as weights of a neural network [7; 8; 9; 10; 11; 12; 13; 14]. 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 [15].

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.*, Fourier frequency mapping) allows coordinate-MLPs to learn high-frequency information more effectively [7; 8]. This observation was recently characterized theoretically by Tancik *et al.* [16], 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 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. In summary, the contribution of this paper is three-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. Promising empirical reconstuction performance is obtained on both 1D and 2D signals using our proposed embedding function in conjunction with coordinate MLPs.

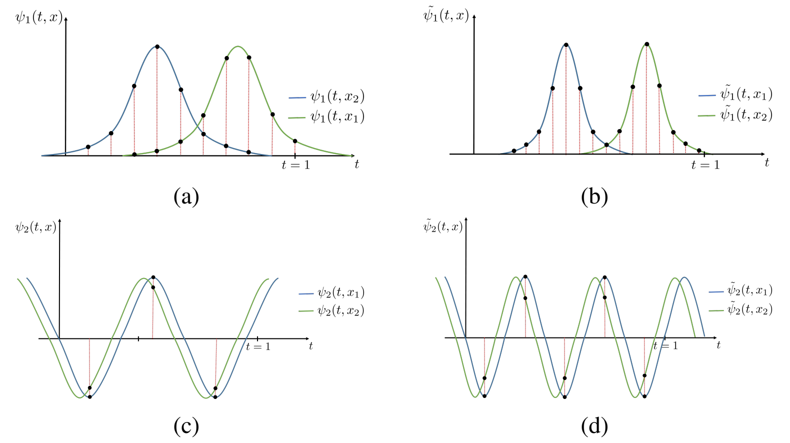


Figure 1: Overview of the proposed positional encoding scheme. Positions are encoded as equidistant samples from shifted basis functions (embedders). The shifts are determined by the corresponding coordinate positions we are wanting to embed. In (a) and (b), x_1 and x_2 are encoded as samples from shifted 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 Sec. 2). 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 Positional embedding: 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. The rationale for choosing a linear leaner is two-fold: a) 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. b) The last layer of a coordinate-MLP is typically linear. Hence, the output of the penultimate layer can be considered as a positional embedding of a linear model. Thus, we intend to study the preferred characteristics of the penultimate layer and then inject those properties into the positional embedding layer with the hope of achieving better results.

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.

2.1 Rank of the embedded representation

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

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

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:

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

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 $\Psi : \mathbb{R} \to \mathbb{R}^d$. 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.*

2.2 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 Sec. 2.1, 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 $\psi : \mathbb{R}^2 \to \mathbb{R}$ 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.

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

2.3 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 practical implementations, 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.*

3 Analysis of possible embedders

Although our derivations in Sec. 2 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 [17] 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 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 then 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.

Sine embedder. Consider $\psi(t, x) = \sin(f(t - x))$ for an arbitrary fixed 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 demonstrates similar properties in terms of the stable rank and the distance preservation (see Fig. 3).

Gaussian embedder. We define the Gaussian embedder as $\psi(t, x) = \exp(-\frac{\|t-x\|^2}{2\sigma^2})$ 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.

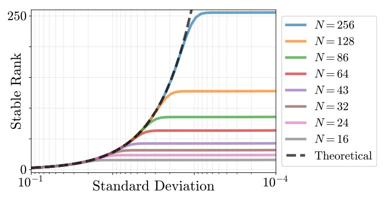


Figure 2: Stable rank of the Gaussian embedder against the standard deviation for different number of samples. The dash line is the theoretical stable rank $\frac{1}{2\sqrt{\pi}\sigma}$

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

(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.

3.1 Connection to the Random Fourier Features

The prominent way of employing Fourier frequency mapping is via Random Fourier Features (RFF) mapping, 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:

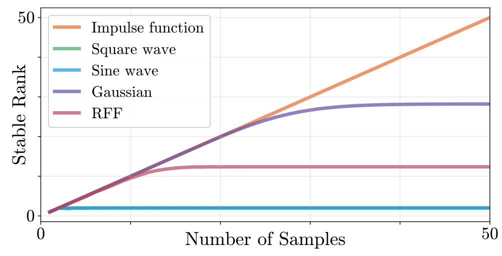
Proposition 3. *Let the RFF embedding be denoted as* $\gamma(x) = [\cos(2\pi\mathbf{b}x), \sin(2\pi\mathbf{b}x)]$, *where* $\mathbf{b}$ *are sampled from a Gaussian distribution. When the embedding dimension is large enough, the stable rank of RFF will be* $\min(N, \sqrt{2\pi}\sigma)$, *where* N *is 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. 6, the stable rank of RFF inceases linearly with the number of samples until it gets saturated at $\sqrt{2\pi}\sigma$. This gives us a relationship between RFF and the Gaussian embedder: Let σ_g and σ_f be the standard deviations of the Gaussian embedder and RFF, respectively. When their stable ranks are equal, $\frac{1}{2\sqrt{\pi}\sigma_g} = \sqrt{2\pi}\sigma_f$ (from Proposition 2 and 3). This implies that when $\sigma_g\sigma_f = \frac{1}{2\sqrt{2}\pi}$, these two embeders are equivalent in terms of the stable rank and distance preservation (observe Fig. 6 when $\sigma_g = 0.01$ and $\sigma_f = 0.1$).

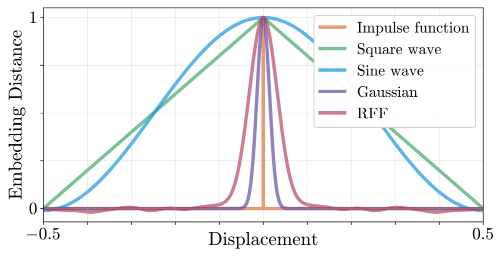
Also, a common observation with RFFs is that when σ_f is too low, the reconstruction results are overly smooth and if σ_f is too high, it gives noisy interpolations [16]. This observation directly correlates with our theory. See in Fig. 6 that when the standard deviation increases, the stable rank increases and distance preservation decreases. Similarly, When the standard deviation is too low, the stable rank decreases while distance preservation increases.

4 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.



(a)



(b)

Figure 3: Quantitative comparison of (a) the stable rank and (b) the distance preservation of different embedders and random Fourier features. 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. Note that as predicted in theory, 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.

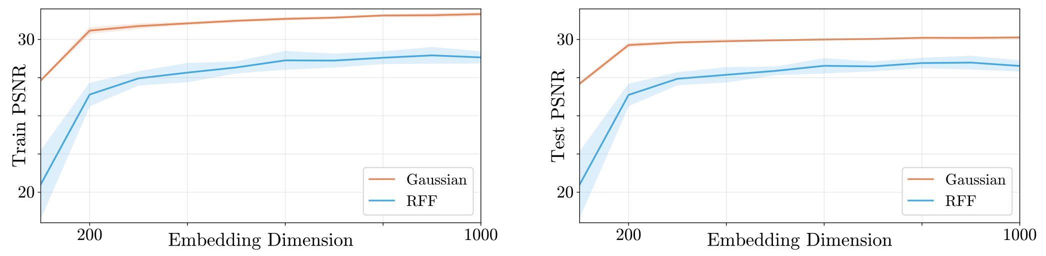


Figure 4: Stability of the performance at different embedding dimensions when encoding a 1-D signal. Shaded areas correspond to two standard deviations across 10 experiments. As illustrated, the Gaussian embedder demonstrates much stable performance, especially at lower embedding dimensions.

4.1 The Gaussian embedder vs RFF

The proposed positional embedding scheme is a deterministic process, *i.e.*, there is no ambiguity in performance against hyper-parameters. In comparison, RFF samples frequencies randomly, leading to unstable performance, especially when the embedding dimension is low. In order to verify this, we use a 1D signal (a stripe from the popular Pepper image) as the target signal and train a linear network with a single layer with both embeddings. Following Sec. 3.1, we choose the standard deviations $\sigma_g = 0.005$ and $\sigma_f = \frac{1}{2\sqrt{2}\pi\sigma_g}$. We run the experiment 10 times with random initializations and obtain the means and the standard deviations for different embedding dimensions. As reported in Fig. 4, the variance of performance in the Gaussian embedder is much smaller compared to RFF. Note that the performance variance of the RFF decreases as the embedding dimension increases. In particular, this means that the ambiguity of performance in RFF can be reduced with a sufficiently large embedding dimension. However, at lower embedding dimensions, the Gaussian embedder demonstrates less ambiguity and better performance.

4.2 Performance of different embedders

We empirically compare the performance of different embedders and verify that the theoretically predicted properties hold in practice. We use a single-layer linear MLP for this experiment and choose the embedding dimension to be 10000. Further, we pick ten random rows from the Pepper image as our targets and measure the average performance. We train the model with each embedder for 4000 epochs using the Adam optimizer with a learning rate of $1e^{-4}$.

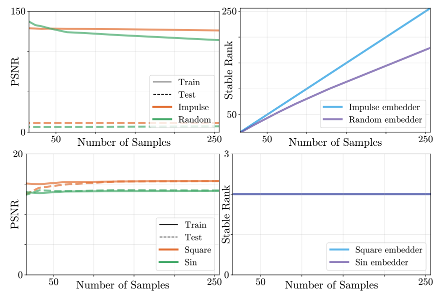


Figure 5: Performance and stable ranks of different embedders against the number of sampled points.

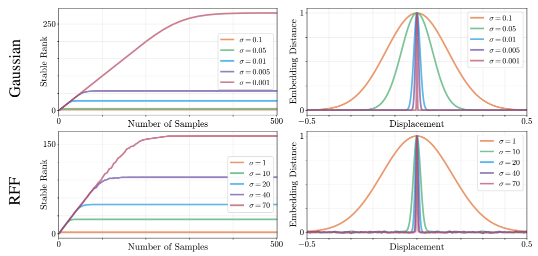


Figure 6: The stable rank (left column) and distance preservation (right column) of the Gaussian embedder and RFF across different standard deviations.

Impulse & Random embedder: We implement the impulse embedder as a one-hot encoder. The results are shown in Fig. 5. It is clear from the theory due to the unbounded stable rank (or equivalently, the unlimited bandwidth) of the impulse embedder, the train PSNR should be high and the test PSNR should be poor. To further validate our theory, we employ a random noise as the embedder and compare its performance. Note that the bandwidth of a random noise is extremely high and as expected, its behavior is similar to the impulse embedder.

Square & Sin embedder: As discussed in Sec. 3, these two embedders contain low stable ranks that are approximately upper-bounded at 2. Consequently, we can observe a low training PSNR in Fig. 5. Interestingly, we also observe a lower test PSNR, since extremely small stable ranks cause overly smooth generalizations.

Gaussian embedder: The experimental results for the Gaussian embedder is depicted in Fig. 7. As per Proposition 2, the standard deviation σ of the Gaussian embedder can tune the trade-off between the rank and distance preservation. For each number of samples N, 50 samples of σ are chose log-linearly from $[10^{-4}, 10^{-1}]$. Recall that the stable rank of the Gaussian embedding matrix is $\min(N, \frac{1}{2\sqrt{\pi}\sigma})$, and a higher rank can achieve good memorization (a higher training PSNR). However, from the experimental results, it is evident that we do not need full rank to have a good enough PSNR on real signals since real signals contain redundancies. Experimentally, training PSNR generally hits a peak near $\sigma = \frac{1}{2N}$ (indicated by star key points). Similarly, the test PSNR depends on distance preservation. The experiment results show that the test PSNR decreases as σ get very small. This is intuitive, since when σ is too small, the output is overly smoothened. Then, for a small interval of σ, the test PSNR keeps constant before it begins to drop drastically. The reason is obvious, as when σ is very high, the distance preservation is hampered. To avoid this, we need the distance to be preserved *at least* between the two nearest sampled points from the input signal. Therefore, for N equally spaced training samples, we need distance to be preserved in an interval $l \approx \frac{1}{2N}$. From the Proposition 2 we know that $\sigma = \frac{1}{4N\sqrt{k \ln 10}}$ where k is an empirically chosen threshold. For our experiment we choose $k = 1.6$, thus, $\sigma = \frac{1}{4N\sqrt{1.6 \times \ln 10}}$. This corresponds well with the rightmost key points on the test PSNR plot. The leftmost key points on test PSNR plot is $\sigma = \frac{10}{4N\sqrt{1.6 \times \ln 10}}$, which means the distance is preserved over an interval of five nearest points on either side.

4.3 Reconstruction of 1-D signals

We use three random rows from an image to test the reconstruction ability of the Gaussian embedder for a 1-D signal. The training data are sampled with an interval of one. We use a one-layer linear network and train for 2000 epochs. The results are shown in Fig. 8. As illustrated, for a very large σ, the stable rank is too low, leading to poor memorization and overly smooth generalization. When σ is too small, the memorization is better, but the generalization is poor. A mid-range σ learns the mapping best. A notable advantage of the Gaussian embedder is that it will output zeros outside the learning range instead of meaningless random numbers.

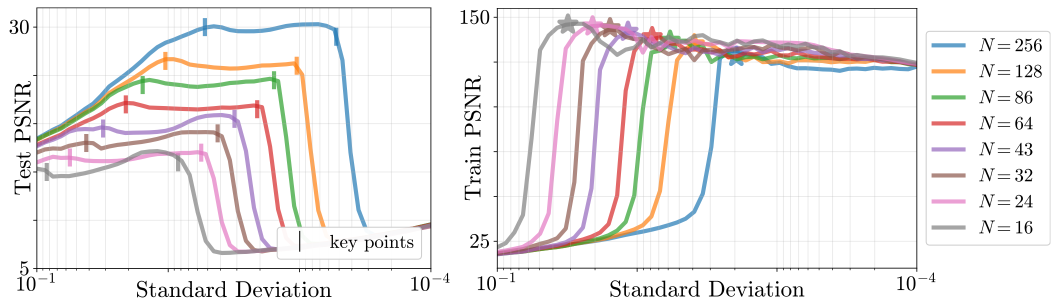
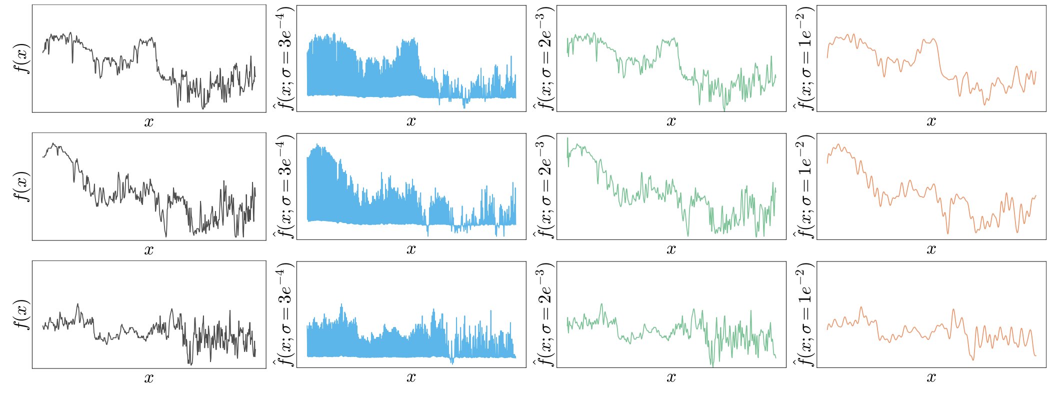


Figure 7: Performance of the Gaussian embedder against the standard deviation over different number of input samples.

σ	0.1	0.5	0.05	0.025	0.012	0.006	0.003	0.001	0.0007	0.0003	0.0002	0.0001
Train PSNR	17.59	19.25	21.83	24.98	29.54	52.77	125.42	113.22	113.62	115.76	115.38	115.41
Test PSNR	17.57	19.21	21.77	24.75	28.41	31.63	30.87	30.00	19.21	12.17	12.81	12.21

Table 1: Quantitative comparison of 1D signal reconstruction performance of the Gaussian embedder against the standard deviation (σ). Memorization ability of the model increases with σ. In contrast, the model generalizes better for a mid-range σ. We use a 4-layer MLP with Gaussian positional embedding for this experiment.

We also conduct an experiment where we pick 1D signals, and measure the train and test PSNRs across varying σ, using a 4-layer non-linear MLP. The tests are run for 10 signals and the average numbers are reported in Table 1. As shown, the training PSNR monotonically decreases against σ, implying worsening memorization. In contrast, extremely low σ and high σ corresponds to low test PSNR due to overly smooth generalization and poor generalization, respectively. A mid-range σ demonstrates best results.



(a) Ground truth (b) $\sigma = 0.0003$ (c) $\sigma = 0.002$ (d) $\sigma = 0.01$

Figure 8: Qualitative comparison of reconstructing 1-D signals using the Gaussian embedder across different standard deviations. A too small σ demonstrates poor generalization, a too high σ gives over-smooth generalization, while a mid-range σ yields better results.

4.4 Extension to 2-D signals

A seemingly critical disadvantage of our embedding mechanism is that when working with higher dimensions, the embedding dimension should grow exponentially (in order to facilitate the dense sampling of embedders in higher dimensions). However, this can be alleviated using separable functions as the embedders. A straightforward example for this is the Gaussian embedder. Recall that high dimensional Gaussians with diagonal covariance matrices are separable along axes. Therefore, we can employ 1-D embedders along each dimension and then concatenate the embedder outputs for

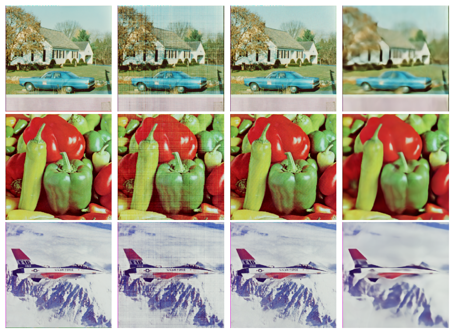


Figure 9: Qualitative results of reconstructing 2-D signals using the Gaussian embedder across different standard deviations (best viewed in zoom). Here, we only sample along two directions.

Embedding	Image type	Train PSNR	Test PSNR
No PE	Natural	20.42	20.39
Sin	Natural	22.94	22.63
RFF	Natural	34.53	26.03
Gaussian	Natural	34.52	27.19
No PE	Text	18.49	18.49
Basic	Text	20.52	20.49
RFF	Text	38.39	31.71
Gaussian	Text	36.93	30.65
No PE	Noise	10.82	10.78
Basic	Noise	10.83	10.78
RFF	Noise	17.60	9.20
Ours	Noise	11.41	10.55

Table 2: Quantitative comparison between no embedding, sin embedding, RFF, and Gaussian embedding in 2D image reconstruction.

each position. As a result, the embedding dimension only increases linearly with the dimension of the input signal. However, there is an associated drawback with this method, which we will discuss next.

Consider sampling a 1-D separable embedder along x and y axes separately and concatenating them to obtain the embedding for each (x, y) point of a 2-D signal. Also, denote the ground truth signal as $I(x, y)$. Then, using a linear model we have, $I(x,y) \approx \mathbf{w}_x^T \Psi(\mathbf{x}) + \mathbf{w}_y^T \Psi(\mathbf{y})$. This can be written in the matrix form as,

$$\begin{bmatrix} I(1,1) & \dots & I(N,1) \\ \vdots & \ddots & \vdots \\ I(1,N) & \dots & I(N,N) \end{bmatrix} \approx \begin{bmatrix} \mathbf{w}_x^T \Psi(x_1) & \dots & \mathbf{w}_x^T \Psi(x_N) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_x^T \Psi(x_1) & \dots & \mathbf{w}_x^T \Psi(x_N) \end{bmatrix} + \begin{bmatrix} \mathbf{w}_y^T \Psi(y_1) & \dots & \mathbf{w}_y^T \Psi(y_1) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_y^T \Psi(y_N) & \dots & \mathbf{w}_y^T \Psi(y_N) \end{bmatrix}$$

Clearly, right hand side matrices are of rank one. Therefore, a linear network can only reconstruct an image signal with at most rank 2. However, this drawback can be addressed in most practical cases using a non-linear MLP with a higher number of layers. Therefore, a vital insight to note here is that the advantage of using deeper and non-linear networks to encode signals becomes more significant as the dimensionality of the input signal increases.

A qualitative example is shown in Fig. 9. For this experiment, we use a 4-layer MLP with ReLU activation and only sample along x and y axes separately using a Gaussian embedder to obtain the embedding matrix. We employ 256 neurons in each layer and train for 2000 epochs. For each image, we choose 25% of the total pixels (regularly sampled) as training samples. Similar to 1-D signal reconstruction, using a mid-range standard deviation for the Gaussian embedder works best.

We also conduct a comparison on the 2D image dataset released by by [16]. The dataset consists of natural images, text images, and noise images, where each category contains 32 images. Original images are of 512×512 resolution. A sub-sampled grid of 256×256 pixels is used as training data, and the remaining pixels are used as test data. The MLP consists of three hidden layers, with 256 neurons in each hidden layer followed by ReLU activation. Results are illustrated in Table 2. All reported values are average quantities measured over each category. In contrast to 1D signal reconstruction, the Gaussian embedder demonstrates slightly inferior performance to RFF due to sparse sampling. We conducted an exhaustive hyper-parameter sweep for RFF and Gaussian embedder to obtain optimal performance.

5 Conclusion

In this paper, we develop a novel perspective on positional encoding. In summary, we show that the performance of a positional embedding 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 positional encoding mechanism that can incorporate arbitrary continuous signals as potential embedders, under certain constraints. This allows for a more interpretable and less restrictive way to encode positions that can be used in various computer vision tasks.

REFERENCES

[1] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[5] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[6] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR, 2020.

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[8] Ellen D Zhong, Tristan Bepler, Joseph H Davis, and Bonnie Berger. Reconstructing continuous distributions of 3d protein structure from cryo-em images. *arXiv preprint arXiv:1909.05215*, 2019.

[9] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[10] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020.

[11] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. *arXiv preprint arXiv:2011.10379*, 2020.

[12] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. *arXiv preprint arXiv:2012.03065*, 2020.

[13] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020.

[14] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*, 2021.

[15] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[16] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[17] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21–es, 2007.

[18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[19] Z Dai, Z Yang, Y Yang, WW Cohen, J Carbonell, QV Le, and R Transformer-XL Salakhutdinov. Attentive language models beyond a fixed-length context. arxiv 2019. *arXiv preprint arXiv:1901.02860*.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[21] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[22] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*, 2018.

[23] Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*, 2019.

[24] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. *Advances in Neural Information Processing Systems*, 32:12081–12091, 2019.

[25] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

[26] Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *International Conference on Machine Learning*, pages 6327–6335. PMLR, 2020.

[27] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

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

[29] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[30] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[31] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019.

[32] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[33] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[34] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*, 2019.

[35] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[36] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[37] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.