Learnable Fourier Features for Multi-Dimensional Spatial Positional Encoding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Attentional mechanisms are order-invariant. Positional encoding is a crucial component to allow attention-based deep model architectures such as Transformer to address sequences or images where the position of information matters. In this paper, we propose a novel positional encoding method based on learnable Fourier features. Instead of hard-coding each position as a token or a vector, we represent each position, which can be multi-dimensional, as a trainable encoding based on learnable Fourier feature mapping, modulated with a multi-layer perceptron. The representation is particularly advantageous for a spatial multi-dimensional position, e.g., pixel positions on an image, where L_2 distances or more complex positional relationships need to be captured. Our experiments based on several public benchmark tasks show that our learnable Fourier feature representation for multi-dimensional positional encoding outperforms existing methods by both improving the accuracy and allowing faster convergence.

1 Introduction

Attentional mechanisms are a central component in many deep architectures [1, 20], which allow a model to selectively focus on specific information in the context. Transformer [32] and its many variants, such as [23, 32, 13, 3], which are solely based on attentional mechanisms, have advanced the state of art on many tasks that involve data with inherent temporal and spatial orders, e.g., machine translation [32], image generation [13], and object detection [3].

In contrast to recurrent [11, 28] or convolutional architectures [15], which automatically capture the ordinal information as computation progresses based on sequential or spatial dependencies, attentional mechanisms are order invariant. It allows a model to directly access information at an arbitrary position in a sequence or space. The lack of ordinal information in the model is not an issue when attentional mechanisms are combined with a recurrent or convolutional architecture [1, 20]. However, it is crucial for Transformer-alike models where the entire model is built based on attentional mechanisms.

To capture positional information in the data, e.g., the token position in a sentence or the pixel coordinates in an image, *positional encoding* has been introduced [9, 32], which a position in a one or two-dimensional space is mapped to a vector space by either learning or heuristics-based approaches. The representation of an input, by combining both its positional encoding and content representation, e.g., word embeddings, then participates in downstream computation for attentional mechanisms. The original Transformer model uses a fixed sinusoidal encoding with predefined wavelengths [32]. However, the predefined features lack flexibility and may not capture important position information in a task-dependent manner. To encode positions in a more flexible and data-driven way, position embedding approaches (e.g., one used in BERT [7]) introduce trainable embedding vectors for each (absolute or relative) position. Unfortunately, this data-driven approach comes at the cost of

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

introducing a large amount of extra learnable parameters proportional to sequence lengths times the hidden dimension size. Moreover, it is non-trivial to apply position embedding to problems with variable sequence lengths.

In this paper, we consider the problem of designing a position encoding for multi-dimensional spatial positions, such as pixel positions in an image or object bounding boxes in a spatial structure such as UIs. Existing methods typically use the position encoding or embedding approach to encode each dimension independently and then combine the resulting vector representations via concatenation, e.g., [23, 3, 8]. Unfortunately, those approaches suffer from several drawbacks. First, they are not able to capture desired positional similarity on an image, such as L_2 distance or more complex positional relationships. Moreover, since the sequence length grows exponentially to the input dimension, the position embedding approach introduces large overhead in 2D and could be infeasible scaling to a higher dimensional space. Finally, special treatments are needed to adjust the position embedding when the test image sizes differ from training, such as bicubic interpolation used in DeiT [31] or Vision Transformer [8].

The main contributions of our work are as follows. We design a novel positional encoding method that learns a function to map multi-dimensional positions into a vector space. The function extracts position information based on a set of Fourier features and passing them to an MLP. The encoding function is *learnable* and is initialized in such a way that the inner products of our positional encodings approximate Euclidean distances. The inductive bias can be desirable in a 2D or higher-dimensional space and by learning from the data, the representation can be adapted to a specific problem. Since our method learns an encoding function instead of embedding vectors for each position, it is naturally *inductive* and can handle test samples with arbitrary length. Our method is *parameter-efficient*, in the sense that the number of parameters do not grow with sequence length. To allow complex positional relationships, our representation is also *composable* by encoding each subset of dimensions, in a multi-dimensional space, using a shared learnable Fourier features. We evaluate our method on a number of tasks where Transformer-based models have been used for problems with multi-dimensional positions, including image generation [13] and object detection [3], which both involve 2D positions in images. We also evaluate our method on natural language generation in graphical user interfaces, which involve modeling a spatial structure of UI objects on the screen, where each object is characterized 4-coordinate values [17]. These experiments show that our positional encoding method consistently outperforms existing methods by both improving accuracy and accelerating learning.

2 Background

2.1 Positional Encoding

In Transformer models, the self-attentional mechanism determines the strength between each pair of items based on the dot product similarity of their vector representations, which are derived from an item's content embedding and positional encoding [32]. Although positional encoding (PE) does not function alone in determining the attention strength, the benefit of having the inductive bias of positional relevance in the PE is evidenced by the success of the Sinusoidal positional encoding originally proposed in Transformer [32] (Equation 1).

$$PE(p, 2d) = \sin \frac{p}{10000^{2d/D}}; PE(p, 2d+1) = \cos \frac{p}{10000^{2d/D}} \tag{1}$$

which encodes a scalar position, p, using Sinusoidal functions with different constant frequencies for each dimension, d, of a D-dimensional encoding vector. The dot product of this encoding representation naturally captures positional similarity in a 1D sequence in a parameter-free fashion.

The other category of approaches for PE is to treat each position as a discrete token that can then be uniquely represented as a learnable embedding vector [8, 9, 13, 7]. The approach can capture arbitrarily complex relationships between positions by learning from data, but it can be difficult to generalize for positions that are rarely encountered during training. For example, the heatmap map in Figure 1 shows the positional similarity learned by a Transformer model for a machine translation task on En-De WMT32k [32]. Towards the diagonal, i.e., positions that are closer, there tends to be higher similarity because each token attends to itself the most. However, the trend is diffused for large positions, i.e., when a sequence is long,

because fewer training examples have long sequences. For what is followed, a model will not be able to correctly represent large positions in a long sequence at training and test time.

There has been extensive work in extending positional encoding for difficult modeling tasks, e.g., handling long-range sequences [6, 36] or tree structures [29, 34]. Our work is related to the effort of using a continuous function instead of embedding retrieval for modeling positions, e.g., previous work [33] using complex embedding functions to model 1D positions. In this work, we focus on encoding multi-dimensional spatial positions using learnable Fourier features to bring in inductive bias. Our work is different from the body of work on relative positional encoding, which directly represents pairwise positional relation between query and key [27, 12, 2, 26]. Because there are $O(N^2)$ of pairwise relations for N positions, relative positional attention is only feasible for a small range, e.g., within a clip distance or local range. Another form of relative PE is to average around neighbor positions [22]. In our work, we focus on representing individual multi-dimensional spatial positions such that these representations achieve desirable pairwise relation later during attention computation. More recently, it has been shown that position encoding in 1D space can be learned as a Neural ODE system [19]. However, their approach cannot be extended to 2D or higher-dimensional problems.

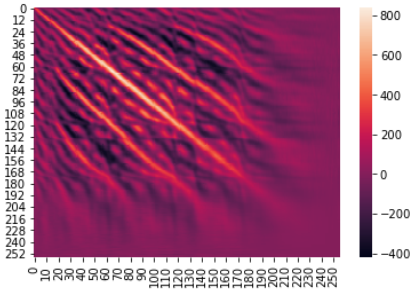
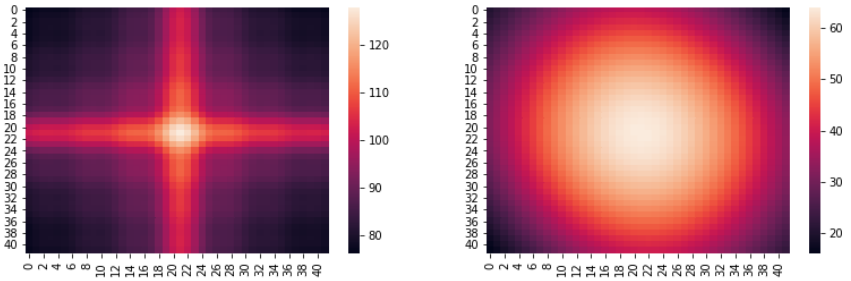


Figure 1: The heatmap shows the dot product similarity of positional embeddings learned by a Transformer model for the En-De WMT32k machine translation task.

2.2 Encoding Multi-Dimensional Spatial Positions

A common approach for positional encoding for a 2D problem is to encode each positional dimension (vertical and horizontal) independently using either sinusoidal (Equation 1) or direct embedding-based methods, and then concatenate the two to form the final positional encoding [23, 3, 13, 8]. Although the approach of sinusoidal concatenation allows the model to capture the positional (spatial) relationships orthogonally along each axis, the similarity decays much faster along other directions (see Figure 2(a)), which ideally should decay at the same rate along all the directions for modeling L_2 distances.



(a) Sinusoidal-concatenation encoding.

(b) 2D Fourier feature positional encoding.

Figure 2: The similarities of the center position to the rest positions on the 2D space, based on the dot product between their positional encoding of each approach.

While concatenating learned embedding has the capacity to model complex spatial relations between positions, they can be difficult to generalize. It is even brittle for addressing problems involving higher-dimensional positions. For example, for modeling spatial structures in UIs [16, 17], recent work takes a collection UI objects as input and the position of each object on the screen involves 4 coordinates: `[top, left, bottom, right]`. The occurrence of unique positions can be much sparser in a multi-dimensional positional space such as bounding boxes. It makes a model difficult to

generalize as there are more chances to encounter unseen positions during testing, as we will show in the experiment. Motivated by these analyses, we intend to develop a positional encoding method for representing a multi-dimensional position by taking into account all the dimensions holistically, which includes incorporating more effective inductive bias, and at the same time allowing learnable representation.

2.3 Fourier Features

The task of mapping data points to a vector space such as their dot product achieves certain distance metric has been extensively investigated in the literature of kernel functions [24, 25, 14, 35, 10, 30].

$$k(x, y) \approx z(x)' z(y)$$

where $x, y \in \mathcal{R}^d$ and $k(x, y)$ is a shift-invariant kernel function; and $z(x)$ and $z(y)$ are feature mapping respectively.

Fourier feature [24, 25] is a common technique to approximate Gaussian kernel, a shift-invariant kernel, with $k(x, y) = \exp(-\frac{\|x-y\|^2}{\gamma^2})$ where $\|x - y\|^2$ is the Euclidean distance between two points, x and y, which each point is a multi-dimensional position in our context. This unique attribute inspired us to represent a multi-dimensional position via Fourier features, which is a basis for our approach for positional encoding.

Random Fourier feature has also been applied in deep learning models, e.g., approximating the attention matrix in Transformer [5]. Recently, adaptive random Fourier features [18] are proposed for better kernel approximation that show improvement on classification tasks. In contrast, we propose learnable Fourier features for spatial positional encoding and integrate the method in various Transformer-based deep architectures that show improvements on multi-dimensional spatial tasks.

3 Learnable Fourier Features Positional Encoding

We propose to learn a position encoding function that maps an M-dimensional position $x \in R^M$ into a K-dimensional feature vector. This K-dimensional vector will then be used in downstream computation for attention mechanisms. The proposed encoding function is composed with the following two components:

Learnable Fourier Features To extract useful features from the input position x, we consider the following feature extraction layer motivated by the idea of Fourier features [24, 25]. Given an M-dimensional position, $x \in R^M$, we acquire a D-dimensional Fourier feature vector representation for the position, $r_x \in R^D$, as follows:

$$r_x = \frac{1}{\sqrt{D}}[\cos xW_r^T \parallel \sin xW_r^T] \tag{2}$$

where $\parallel$ is the concatenation of two vectors. This can also be viewed as the generalization of Sinusoidal position encoding to the multi-dimensional case, while we set W_r, which defines both the orientation and wavelength of Fourier features, as trainable parameters. Since $\cos(a - b) = \cos a \cos b + \sin a \sin b$, we have

$$r_x \cdot r_y = \frac{1}{D}\text{sum}\big(\cos((x - y)W_r^T)\big) := h_{W_r}(x - y).$$

Therefore, vectors in the form of (2) enjoys the shift-invariance property – the inner product of r_x and r_y is a function of $x - y$ and the function is parameterized by W_r. Learning W_r is equivalent to obtaining the most informative function on $x - y$ that can be useful for the downstream task.

In our algorithm, the linear projection W_r are initialized by drawing from a normal distribution

$$W_r \sim \mathcal{N}(0, \gamma^{-2}). \tag{3}$$

When the linear projection weights are drawn in such a way, according to random Fourier features [24, 25], the dot product between two feature vectors, r_x and r_y, approximates the Gaussian kernel over the original positions.

$$r_x \cdot r_y \approx \exp(-\frac{\|x - y\|^2}{\gamma^2}). \tag{4}$$

Figure 2(b) visualizes this representation, which introduce a useful inductive bias of L_2 distances into the model.

MLP layer To feed the representation to the downstream computation, we give the representation additional capacity by modulating the features with a multi-layer percetron:

$$PE_x = \phi(r_x, \theta)W_p, \tag{5}$$

where θ are trainable parameters. W_p are trainable parameters for projecting the representation onto a target dimension of positional encoding for combining with content embedding. Our purpose with MLP here is very different from previous work that uses non-linear transformation such as RNN to capture positional dynamics [21, 19]. These previous works do not handle non-sequential multi-dimensional positions.

The learnable parameters in our position encoding function are W_r for Fourier features and θ, W_p for the MLP layer. The size of these matrices are independent to the sequence length. Further, the position encoding function can be applied to any input position x, so our method can be easily applied when training and testing images have different resolutions. Compared to the previous sinusoidal representation (Equation 1), our representation is learnable and multi-dimensional. Compared to the discrete embedding-based approach, our representation treats each dimension of a position as a continuous value, which alleviates the sparsity issue with discrete positions.

Our representation is applicable for many 2D spatial tasks, e.g., images. For higher-dimensional positions, the positional similarity between positions might be more complicated than L_2 distances. For example, to model the spatial structure of a scene or a UI, given two objects in the structure, x and y, their four coordinate values, $[x_1, x_2, x_3, x_4]$ and $[y_1, y_2, y_3, y_4]$, represent the object's top, left, bottom, and right. The L_2 distance between the two positions $\sum_{i=1}^{4}(x_i - y_i)^2$ will capture neither the minimum nor maximum distance between the two objects, or any vertical or horizontal alignments of them. To address this issue, we hypothesize that complex spatial relationships can be built on top of shift-invariant relations enabled by our positional encoding. Specifically, we can partition a multi-dimensional position into groups, and apply the same encoding pipeline to each group of coordinate values. The process is similar to applying convolution over partitions with the kernel and stride sizes to be the group size. We can then concatenate the output of all the groups to form the final positional encoding. We will elaborate on this use case in the UI modeling experiment (Section 4.3). An implementation of our positional encoder based on tensor operation is detailed in Algorithm 1.

Algorithm 1: Compute the Fourier feature positional encoding of a multi-dimensional position.

Input: A tensor X in the shape of $[N, G, M]$ that represents N positions where each position is in the shape of $[G, M]$ that represents G positional groups and each group has M-dimensional positional values.
Output: PE_X in the shape of $[N, D]$ where D is the depth of the positional encoding.
Hyperparameter: The depth of the Fourier feature dimension $|F|$, the hidden layer dimension $|H|$, and the positional encoding dimension D, and γ.;
Initialization: Initialize learnable weights $W_r \in R^{\frac{|F|}{2} \times M}$ by sampling from $\mathcal{N}(0,\ \gamma^{-2})$;
Initialize $W_1 \in R^{|F| \times |H|}$ and $B_1 \in R^{|H|}$, and $W_2 \in R^{|H| \times \frac{D}{G}}$ and $B_2 \in R^{\frac{D}{G}}$.;

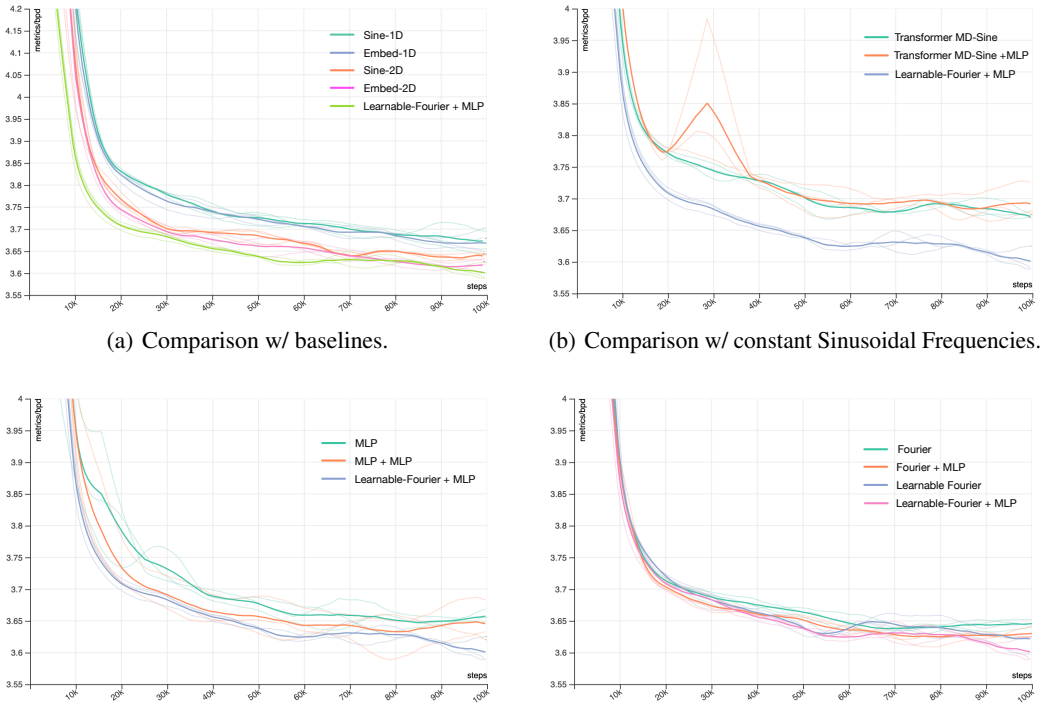
$F \leftarrow \frac{1}{\sqrt{|F|}}[\cos XW_r^T; \sin XW_r^T]$;
$Y \leftarrow \text{Maximum}(0, FW_1 + B_1)W_2 + B_2$;
$PE_X \leftarrow$ Reshape Y into the shape of $[N, D]$;
return PE_X.

4 Experiments

We evaluate our approach on a range of public benchmark tasks using Transformer-based models in comparison with several existing positional encoding methods.

4.1 Image Generation on ImageNet-64

We compare our method with existing positional encoding approaches based on Reformer [13] for the image generation task on the ImageNet 64x64 dataset [4]. Reformer is a Transformer-based



(a) Comparison w/ baselines.

(b) Comparison w/ constant Sinusoidal Frequencies.

(c) Comparison w/ MLP.

(d) Fourier ablation.

Figure 3: Bits per dim (bpd) w.r.t. training steps on the held-out data for Reformer with different positional encoding methods. The plot shows the mean of 3 repeats of experiments for each method.

model that uses locality-sensitive hashing and reversible residual layers to efficiently handle long sequences. Reformer flattens a 64x64 image into a sequence (Length=64x64x3=12,288) in a raster scan red-green-blue order. Reformer as an auto-regressive model predicts the pixel value at each position by attending to previous positions. We equip Reformer with different positional encoding methods.

- *Embed-2D*: Reformer's original positional encoding embeds each spatial dimension separately with two embedding matrices: vertical $[64, 384]$ and horizontal $[64, 384]$.
- *Embed-1D*: The baseline method assigns a learnable embedding to each position in the flattened sequence, which ignores the 2D structure of an image and lets the model to learn positional relations all by itself.
- *Sine-2D* and *Sine-1D*: Similar to Embed-2D and Embed-1D, but they instead encode a position using the original constant sinusoidal formulation (Equation 1).
- *Learnable-Fourier + MLP*: Our method that implements Algorithm 1 using the hyperparameter $|F| = 384$, $|H| = 32$, $D = 768$. We picked these dimensions for our method to have roughly the same number of parameters as Embed-2D, the benchmark of Reformer.

We leave the RGB axis to use the direct embedding as the original Reformer: $[3, 256]$. The concatenation of the pixel position encoding and the RGB index embedding results in an representation that has the same depth (1024) as the one in the original paper, which allows the rest of the model intact.

We follow the experimental procedure as detailed in the Reformer paper. All our experiments used a 6-layer, 8-head-attention Reformer, with $d_{model} = 1024$, $d_{ff} = 4096$, and $n_{heads} = 8$. These models are implemented based on the public Reformer codebase in Trax[1]. The training for each Reformer model is parallelized across 32 TPU v2 cores, and each batch contains 8 sequences (images) on each core.

[1] `https://github.com/google/trax/tree/master/trax/models/reformer`

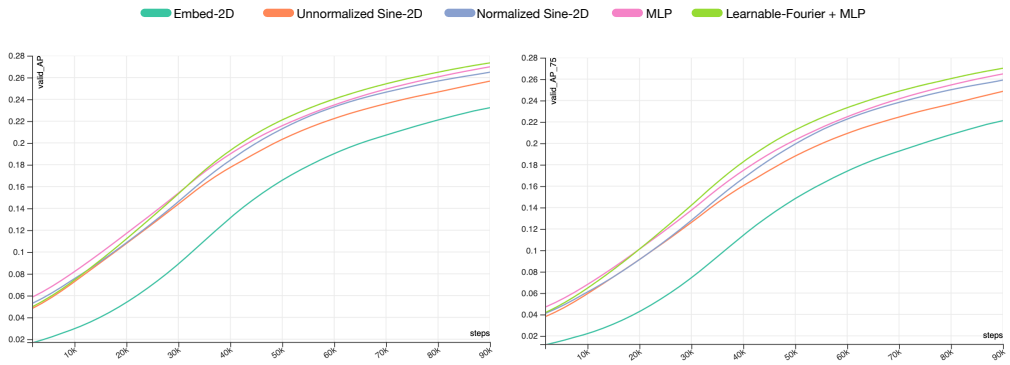


Figure 4: The impact of different positional encoding methods on the DETR model, shown as AP, AP_{75} from left to right, on validation data as training progresses for the initial 100k steps.

As shown in Figure 3(a), our method, Learnable-Fourier + MLP, outperforms all the baselines in terms of both the speed of convergence and achieving better accuracy, i.e., lower bits per dim at the end. The Reformer's original positional encoder, Embed-2D, is the second best. Sine-2D clearly outperforms Sine-1D, and Embed-1D achieves a similar performance as Sine-1D.

To understand how each component in our method contributes, we conduct several ablation studies. We create *Transformer MD-Sine*, a multi-dimensional version of the sinusoidal method (Equation 1), by combining the constant frequencies of two positional dimensions (vertical and horizontal) before applying sinusoidal functions. As shown in Figure 3(b), Transformer MD-Sine performs poorly. It is even worse and unstable when using an additional MLP.

We then wonder whether a simple multi-layer perceptron (MLP) that takes a 2D coordinate and outputs a vector would be able to generate effective encoding. Our experiment shows that MLP does produces usable positional encoding (see Figure 3(c)). However, it is still not as good as our method, even with an additional MLP layer.

Finally, we compare Learnable Fourier versus fixed Fourier features, both with and without an additional MLP layer. We found fixed Fourier features alone (*Fourier*) do not perform as well as those with learnable parameters (Figure 3(d)). When it is enhanced with an additional MLP layer (*Fourier + MLP*), it performs nicely. Overall, Learnable-Fourier + MLP still performs the best. These experiments show that learnability is crucial for our methods to work. See the parameter size for all the model variants in the supplemental material.

4.2 Object Detection

We evaluate our positional encoding method in DETR [3], a recent model that uses Transformer for end-to-end object detection. It uses a Transformer to take the output from a ResNet, which includes 42×42 super pixels. Similar to Reformer, a positional encoding method needs to represent each super-pixel position as part of the input to the Transformer encoder in DETR.

In our experiment, we use the default 6-layer Encoder-Decoder setup in DETR, using all the same hyperparameters, which uses COCO 2017 detection dataset with 118k images for training and 5k for validation. We equip the DETR model with different positional encoding methods, including our method with Learnable-Fourier, Sine-2D Normalized (DETR's default method) and Un-normalized, as well as Embed-2D. All these baseline methods are based on the DETR codebase[2], which are ported into JAX[3], a library for machine learning. The training for each DETR model is parallelized across 64 TPU v3 cores with a batch size of 64 images. We follow the experimental procedure of the DETR paper, and report validation bbox AP and AP_{75}. In this experiment, we focus on how different positional encoding methods impact the convergence of learning.

As shown in Figure 4.2, Learnable-Fourier allows the fastest convergence among all the methods. We see a bigger gain of our method on the AP_{75} metric that requires more precise matches between

[2] `https://github.com/facebookresearch/detr/blob/master/models`

[3] `https://github.com/google/jax`

ground-truth and predicted bounding boxes. The Sinusoidal method performs better when position values are normalized. Normalization is valuable because of random resizing and cropping during image augmentation in DETR. Embed-2D performs poorly compared to the rest methods as it does not leverage position normalization in DETR.

4.3 Widget Captioning in UIs

Positional Embedding	BLEU-1	BLEU-2	ROUGE	CIDEr	METOER	SPICE
SOTA [17]	44.9	32.2	44.7	97.0	31.7	17.6
Embed-4D	45.2	31.9	45.0	97.0	31.7	17.3
MLP	34.0	23.5	33.7	70.3	23.7	10.2
Sine-4D	44.9	31.9	43.9	94.9	31.0	16.7
Learnable-Fourier-2/2	44.9	31.6	44.3	95.3	31.6	17.7
Fixed-Fourier+MLP-1/4	45.0	32.1	44.2	95.4	31.2	17.1
Fixed-Fourier+MLP-2/2	46.1	32.5	45.8	100.2	32.5	18.4
Fixed-Fourier+MLP-4/1	45.5	32.1	45.1	97.2	31.7	17.6
Learnable-Fourier+MLP-1/4	45.6	32.7	45.2	99.1	32.2	17.1
Learnable-Fourier+MLP-2/2	46.1	32.7	45.9	98.0	**32.6**	**17.9**
Learnable-Fourier+MLP-4/1	**46.8**	**33.4**	**46.1**	**100.7**	32.4	17.8

Table 1: The accuracy of different positional encoding methods on the widget captioning test set. SOTA shows the results from the original paper, which is reproduced by Embed-4D in our experiment.

So far, we have investigated tasks that handle 2D positions in an image. In this experiment, we investigate even higher-dimensional positions. In a widget captioning task [17], the model is trained to generate natural language description of widgets in graphical user interfaces, e.g., buttons and icons. A significant part of the model is to encode a UI screen structure, which contains a collection of 2D objects of different sizes, using a Transformer encoder. To represent an object, the original model assigns a learnable embedding vector to each coordinate value of its bounding box: (left, top, right, bottom), and four embedding vectors then jointly represent a bounding box on the screen. We refer this baseline as *Embed-4D*.

Because there is no obvious distance metrics between bounding boxes, we hypothesize that an appropriate metric can be learned on top of L_2 distances of specific dimensions. To do so, We evaluate three different partitions of bounding box dimensions, and use our method to encode each group in parallel as detailed in Algorithm 1: Learnable-Fouier+MLP-1/4 treats all the 4 coordinates `[(top, left, bottom, right)]` as one group; Learnable-Fourier+MLP-2/2 splits the 4 coordinates into 2 groups `[(top, left), (bottom, right)]`; and finally Learnable-Fourier+MLP-4/1 encodes 4 groups of 1 coordinate value `[(top), (left), (bottom), (right)]`. We also add the Sinusoidal approach in the comparison that represents each positional dimension separately and then use their concatenation to represent a bounding box (referred as *Sine-4D*).

We use the same model architecture and hyperparameters of the strongest model, *Pixel+Local+Context*, as the original paper [17], and built our experiment based on the public codebase of widget captioning[4]. Specifically, the screen encoder uses a 6-layer, 8-head Transformer with a hidden size of 128. We train all the models to 100k steps with Adam optimizer and a scheduled learning rate detailed the original paper. All the models converged within 12 hours using 4 V100 GPU.

All the results are acquired by applying each model on the test dataset, based on the same set of captioning metrics. As shown in Table 1, our method outperforms the benchmark method Embed-4D (#Params=5.11M) with a large margin even though our method uses fewer parameters (#Params=5.07M), particularly on BLEU-1, BLEU-2, ROUGE and CIDEr, which clearly advanced the state of art for this task. Interestingly, both Learnable-Fourier+MLP-2/2 and Learnable-Fourier+MLP-4/1 outperform Learnable-Fourier+MLP-1/4, which indicate that more complex distances needed to be modeled in this task. Compared to Embed4D, T-tests (over 3 runs of each model) show the gain of Learnable-Fourier+MLP 4/1 is statistically significant ($p < 0.05$) on BLEU-1, ROUGE, CIDEr and METOER; Learnable-RF 2/2 achieves significance ($p < 0.05$) on BLEU-1, ROUGE and METOER. For the two champion conditions, i.e., Learnable-Fourier+MLP-4/1 and 2/2, we found on most metrics there is no statistical significance between their performance ($p > 0.05$).

[4] `https://github.com/google-research/google-research/tree/master/widget_caption`

Learnable-Fourier+MLP-4/1 outperforms 2/2 only on CIDEr with marginal statistical significance ($p = 0.042$).

We also included a few ablation studies in this experiment. One variant is to fix Fourier features but still include MLP. In this group (Fixed-Fourier+MLP-*), Fixed-Fourier+MLP-2/2 clearly performs the best across all the metrics. Overall, it seems that Learnable-Fourier+MLP still has advantages over the fixed one on most cases. We then look at Learnable-Fourier but without using MLP. Learnable-Fourier-2/2 seems to perform worse than its counterpart in the other groups on every metric. This indicates that MLP is a crucial component, which resonates with our observations in the previous two experiments with image-related tasks. Lastly, although using MLP alone as the encoding function seems competitive in previous two image experiments, it performs poorly in this experiment.

5 Discussion

One clear trend that emerges from our experiments is that positional encoding methods that treats an image as a flatterned sequence (Embed-1D or Sine-1D) do not perform well, even though the model is given a great capacity to learn these positional relations. We also observe that taking a multi-dimensional position holistically often performs better than representing each dimension separately and then concatenating these representations. We found it important to use the multi-layer perceptron (Equation 5) to modulate feature mapping before positional encoding is mixed with content embedding. We got mixed results for using MLP alone as the positional encoding function, which performs competitively on the image tasks but poorly on the UI modeling task that involves sparse spatial structures. It seems that it is not necessary to have a large random feature dimension to achieve good results.

Positional Embedding	Seen CIDEr	Unseen CIDEr
Embed-4D	**123.4**	78.5
Sine-4D	121.3	76.4
Learnable-Fourier+MLP-4/1	**123.4**	**82.2**

Table 2: The accuracy for widgets with seen and unseen positions.

To understand how different positional encoding methods can generalize to unseen positions, we analyze the widget captioning results. There are coordinate values rarely or never seen in the training set (Figure 5). There are 1867 widgets with Seen positions and 2692 with Unseen positions from the test set. Table 2 shows that our method generalizes to unseen positions significantly better than baselines. One direction that deserves further investigation is how the interaction between positional encoding and content embedding should be taken into account for the design of a positional encoding function.

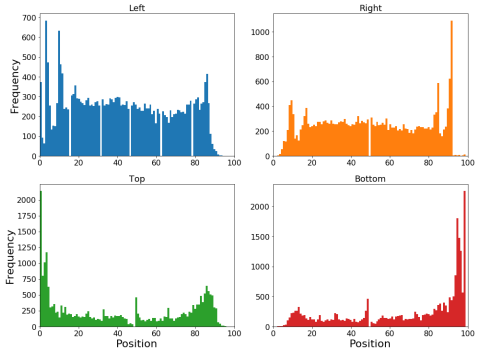


Figure 5: Widget position distributions for each dimension in the training set. There are positions rarely occurred in the training set.

6 Conclusion

We present a novel approach for positional encoding based on learnable Fourier features. We evaluate our approach on a range of multi-dimensional spatial tasks, including image generation, object detection, and sparse spatial structure modeling in user interfaces, which show that our positional encoding consistently outperforms the benchmark methods.

Broader Impact

Our work proposes a general technique for positional encoding in attention-based models. The aspects of ethical and societal impacts are not applicable for this research.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[4] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019.

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, and Adrian Weller. Masked language modeling for proteins via linearly scalable long-context transformers, 2020.

[6] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 06–11 Aug 2017.

[10] Raffay Hamid, Ying Xiao, Alex Gittens, and Dennis DeCoste. Compact random feature maps. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, 2014.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[12] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, and Douglas Eck. An improved relative self-attention mechanism for transformer with application to music generation. *CoRR*, abs/1809.04281, 2018.

[13] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[14] Quoc Le, Tamas Sarlos, and Alexander Smola. Fastfood: Approximate kernel expansions in loglinear time. *30th International Conference on Machine Learning, ICML 2013*, 2014.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989.

[16] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[17] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5495–5510, Online, November 2020. Association for Computational Linguistics.

[18] Yanjun Li, Kai Zhang, Jun Wang, and Sanjiv Kumar. Learning adaptive random features. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4229–4236, Jul. 2019.

[19] Xuanqing Liu, Hsiang-Fu Yu, I. Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *ICML*, 2020.

[20] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[21] Masato Neishi and Naoki Yoshinaga. On the relation between position information and sentence length in neural machine translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 328–338, Hong Kong, China, November 2019. Association for Computational Linguistics.

[22] Andrei Nicolicioiu, Iulia Duta, and Marius Leordeanu. Recurrent space-time graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[23] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[24] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1177–1184, 2007.

[25] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1313–1320, 2008.

[26] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[27] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[28] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.

[29] Vighnesh Leonardo Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In *NeurIPS 2019*, 2019.

[30] Yitong Sun, Anna Gilbert, and Ambuj Tewari. But how does it work in theory? linear svm with random features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3383–3392, 2018.

[31] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[33] Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. *CoRR*, abs/1912.12333, 2019.

[34] Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. Self-attention with structural position representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1403–1409, Hong Kong, China, November 2019. Association for Computational Linguistics.

[35] Jiyan Yang, Vikas Sindhwani, Haim Avron, and Michael Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. *31st International Conference on Machine Learning, ICML 2014*, 1, 12 2014.

[36] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc., 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]