Learnable Fourier Features for Multi-Dimensional Spatial Positional Encoding

Yang Li
Google Research
Mountain View, CA
liyang@google.com

Si Si
Google Research
Mountain View, CA
sisidaisy@google.com

Gang Li
Google Research
Mountain View, CA
leebird@google.com

Cho-Jui Hsieh
UCLA
Los Angeles, CA
chohsieh@cs.ucla.edu

Samy Bengio*
Google Research
Mountain View, CA
bengio@gmail.com

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, 25], which allow a model to selectively focus on specific information in the context. Transformer [38] and its many variants, such as [29, 38, 16, 3], which are solely based on attentional mechanisms, have advanced the state of the art on many tasks that involve data with inherent temporal and spatial orders, e.g., machine translation [38], image generation [16], and object detection [3].

In contrast to recurrent [14, 34, 27] or convolutional architectures [18], 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, 25]. 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 [10, 38], where a position in a

*Currently at Apple.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

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 [38]. 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 [8]) introduce trainable embedding vectors for each (absolute or relative) position. Unfortunately, this data-driven approach comes at the cost of 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 Sinusoidal position encoding with hand-crafted frequencies or learned embedding to encode each dimension independently and then combine the resulting vector representations via concatenation, e.g., [29, 3, 9]. Unfortunately, these approaches, by concatenating the representation of each dimension, are not effective to capture desired positional similarity on an image, such as L_2 distance or more complex positional relationships. While embedding-based approaches have the potential to learn complex positional relationships, since the number of unique positions grows exponentially to the input dimension, the approach incurs large overhead in 2D and could be infeasible scaling to a higher dimensional space. In addition, special treatments are needed to adjust the learned position embedding when the test image sizes differ from training, such as bicubic interpolation used in DeiT [37] or Vision Transformer [9]. To avoid these special adjustments, it is an important for positional encoding to handle unseen positions.

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 [16], object detection [3] and image classification [9], which all involve 2D positions (vertical and horizontal) in images. We also evaluate our method on natural language generation in graphical user interfaces, which involve modeling a sparse spatial structure of UI objects on the screen, where each object is characterized by 4-coordinate values (top, left, bottom, and right) [20]. 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 [38] (Appendix A). 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 [38] (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 [9, 10, 16, 8]. 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 [38]. 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 different modeling tasks, e.g., handling long-range sequences [7, 43] or tree structures [35, 41], or enhancing vision tasks using input-dependent positional encoding [6]. Our work is related to the effort of using a continuous function instead of embedding retrieval for modeling positions. Previous work [40] uses complex embedding functions to model 1D positions. It has been shown that position encoding in a 1D space can be learned as a Neural ODE system [23]. However, their approach cannot be extended to 2D or higher-dimensional problems. More recently, previous work has proposed learnable sinusoidal representations for 1D positions [39] in language tasks. In contrast, we focus on representing 2D or even higher dimensional positions in spatial tasks.

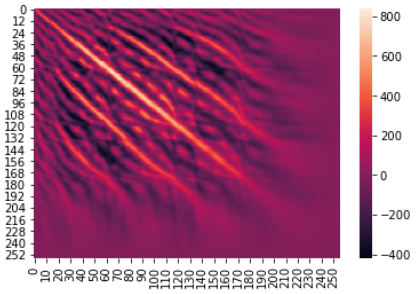


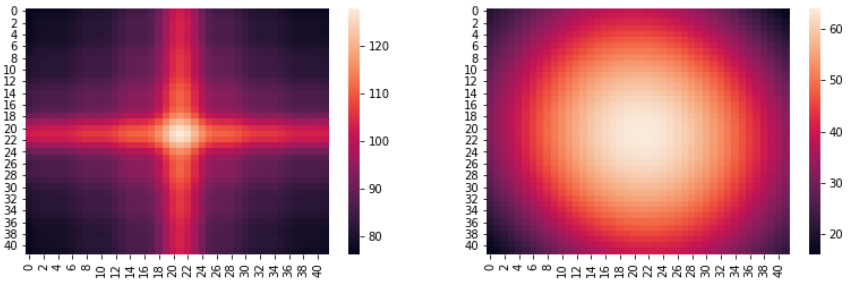
Figure 1: The heatmap shows the dot product similarity of positional embeddings learned by a Transformer model for the En-De WMT32k machine translation task.

Our work is different from the body of work on relative positional encoding, which directly represents pairwise positional relation between query and key [33, 15, 2, 32, 11]. 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, although recent work [24] has achieved linear complexity by approximating relative positional encoding. Because relative positional encoding directly participates in the computation of the attention matrix, instead of addressing the representation of individual input items, it cannot be easily plugged into many existing Transformer architectures. In contrast, our method is fully compatible with many Transformer benchmarks models. In our work, we focus on representing individual multi-dimensional spatial positions such that these representations achieve desirable pairwise relation later during attention computation.

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 [29, 3, 16, 9]. 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.

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 [19, 20], recent work takes a collection of UI objects as input and the positional attribute of each object is its spatial configuration on the screen, which involves 4 coordinate values: `[top, left, bottom, right]`. The occurrence of unique object positions can be sparse, which makes it difficult for a model to



(a) Sinusoidal-concatenation encoding. (b) 2D Fourier feature PE.

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.

generalize. There are positions that are rarely seen during training might occur at test time. 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, and meanwhile enabling effective inductive bias and learnability in the 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 [30, 31, 17, 42, 12, 36].

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

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 features [30, 31] are a common technique to approximate a 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 features have also been applied in deep learning models, e.g., approximating the attention matrix in Transformer [5]. Recently, adaptive random Fourier features [21] have been 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 $x \in \mathcal{R}^M$ 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 [30, 31]. Given an M-dimensional position, $x \in \mathcal{R}^M$, we acquire a D-dimensional Fourier feature vector representation for the position, $r_x \in \mathcal{R}^D$, 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 $W_r \in \mathcal{R}^{\frac{D}{2} \times M}$, 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 the following:

$$r_x \cdot r_y = \frac{1}{D}\text{sum}\big(\cos((x-y)W_r^T)\big) := h_{W_r}(x-y) \tag{3}$$

where $\cdot$ is the dot product. Therefore, vectors in the form of (2) enjoys the shift-invariance property—the dot 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 is initialized by drawing from a normal distribution

$$W_r \sim \mathcal{N}(0,\, \gamma^{-2}). \tag{4}$$

When the linear projection weights are drawn in such a way, according to random Fourier features [30, 31], the dot product between two feature vectors, r_x and r_y, approximates the Gaussian kernel over the original positions.

$$r_x \cdot r_y \approx \exp(-\frac{\|x-y\|^2}{\gamma^2}). \tag{5}$$

Figure 2(b) visualizes this representation, which introduces 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 perceptron:

$$PE_x = \phi(r_x, \theta)W_p, \tag{6}$$

where $\phi(\cdot)$ is the perceptron parameterized by θ. 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 an RNN to capture positional dynamics [26, 23]. 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. Furthermore, the position encoding function can be applied to any input position x, so our method can be easily applied when training and testing involve different positions, e.g., images with 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. Previous work has revealed that using sinusoidal activation functions might suffer optimization problems due to vanishing gradients in extreme cases [28], although we do not observe much difficulty in training our positional encodings.

Our representation is applicable for many 2D spatial tasks, e.g., image-related tasks. For tasks involving 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 natural scene or a graphical user interface, given two objects in the structure, x and y, 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 position. The L_2 distance between the two positions $\sum_{i=1}^{4}(x_i - y_i)^2$ will capture neither the minimum nor the 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 4.4). An implementation of our positional encoder based on tensor operation is detailed in Algorithm 1 in which Equation 2 and 6 are realized in Line 1 and 2.

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 $W_r \in \mathcal{R}^{\frac{|F|}{2} \times M}$ by sampling from $\mathcal{N}(0,\ \gamma^{-2})$;
Initialize $W_1 \in \mathcal{R}^{|F| \times |H|}$ and $B_1 \in \mathcal{R}^{|H|}$, and $W_2 \in \mathcal{R}^{|H| \times \frac{D}{G}}$ and $B_2 \in \mathcal{R}^{\frac{D}{G}}$.;

$F \leftarrow \frac{1}{\sqrt{|F|}}[\cos XW_r^T; \sin XW_r^T]$ (Eq. 2);

$Y \leftarrow \text{ReLU}(FW_1 + B_1)W_2 + B_2$ (Eq. 6) ;

$PE_X \leftarrow$ 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

We compare our method with existing positional encoding approaches based on Reformer [16] for the image generation task on the ImageNet 64x64 dataset [4]. Reformer is a Transformer-based 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.

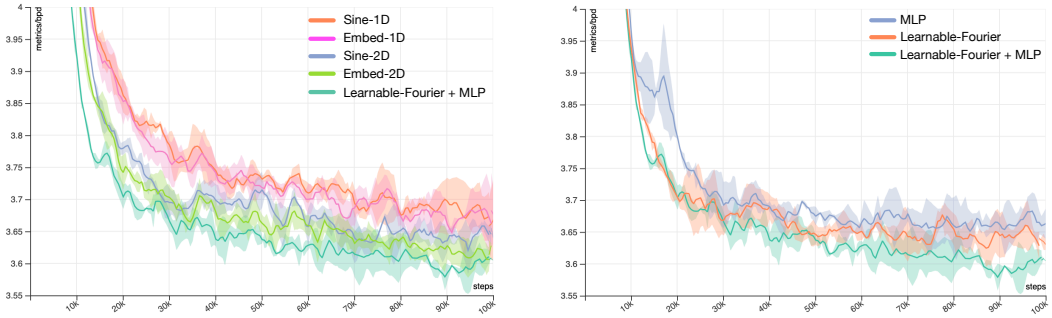
- *Embed-2D*: Reformer's original positional encoding embeds each spatial dimension separately with two embedding matrices: vertical $[64, 384]$ and horizontal $[64, 384]$.
- *Embed-1D*: The baseline method assigns a learnable embedding to each position in the flattened sequence, which ignores the 2D 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[2]. The training for each Reformer model is parallelized across 32 TPU v2 cores, and each batch contains 8 sequences (images) on each core.

As shown in Figure 3a, 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.

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



(a) Comparison w/ baselines.

(b) Comparison w/ individual components.

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 and 95% confidence interval based on 3 repeats of experiments for each method.

To understand how each component in our method contributes, we compare Learnable-Fourier+MLP with its components Learnable-Fourier and MLP alone. MLP takes a 2D coordinate as input and outputs its positional encoding. Our experiment shows that Learnable-Fourier or MLP alone does not perform as good as their combination, Learnable-Fourier+MLP (see Figure 3b). See Appendix C for additional ablation studies.

4.2 Object Detection

We evaluate the proposed positional encoding in DETR [3], a recent model that uses a Transformer for end-to-end object detection. It uses a Transformer to take the output from a ResNet, which includes a feature map with the spatial dimensions of 42×42. Similar to Reformer, positional encoding represents each position as part of the input to the Transformer encoder in DETR. In our experiment, we experiment with the default 6-layer Encoder-Decoder setup in DETR, using the same hyperparameters, on COCO 2017 object detection dataset [22] with 118k images for training and 5k for validation. We equip the DETR model with different positional encoding methods, including Learnable-Fourier+MLP, Sine-2D (DETR's default method), Embed-2D and MLP. The model implementations are based on the DETR codebase[3], which are ported into JAX[4]. The training for each DETR model is parallelized across 64 TPU v3 cores with a batch size of 64 images for 300 epochs to converge, which took about 3 days. We follow the experimental procedure of the DETR paper, and report accuracy on the validation set.

DETR uses image augmentation in both training and validation. Each image is randomly resized to several specific dimensions with the smaller side of the image at one of the following sizes: 480, 512, 544, 576, 608, 640, 672, 704, 736, 768, and 800. For positional encoding, all image positions are normalized to a range of $(0, 1)$. Normalization is valuable because of random resizing and cropping during image augmentation results images with different sizes. Embed-2D treats each position as a discrete value, and all the methods except Embed-2D leverages position normalization. As shown in Table 1, Learnable-Fourier+MLP offers the best performance across all the metrics. Sine-2D and MLP perform competitively while Embed-2D has the worst performance.

To investigate how each encoding generalizes to unseen image sizes, we modify the benchmark by reserving the three largest sizes: 736, 768, and 800 for validation only, which are not seen during training. We also disable position normalization. As a result, there are large positions that are never seen during training, which requires each method to generalize (or extrapolate) to these positions. As shown in Table 2, the benefit of Learnable-Fourier+MLP is more pronounced, and the performance gap between Embed-2D and the other methods is further increased.

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

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

Table 1: The impact of different positional encodings on DETR for object detection.

Method	AP	AP_{50}	AP_{75}	AP_{small}	AP_{medium}	AP_{large}
Sine-2D	40.1	60.4	42.6	18.5	43.6	58.8
Embed-2D	39.3	59.8	41.4	18.7	42.5	57.5
MLP	40.0	60.3	42.2	18.6	43.7	58.1
Learnable-Fourier+MLP	**40.2**	**60.7**	**42.7**	**18.8**	**43.8**	**59.1**

Table 2: The impact of different positional encodings on DETR involving unseen image dimensions.

Method	AP	AP_{50}	AP_{75}	AP_{small}	AP_{medium}	AP_{large}
Sine-2D	38.9	59.6	40.9	17.5	42.5	57.5
Embed-2D	36.6	58.2	37.7	15.9	40.0	55.3
MLP	38.6	59.5	40.3	17.1	42.1	57.1
Learnable-Fourier+MLP	**39.5**	**60.0**	**41.6**	**18.9**	**43.0**	**58.0**

4.3 Image Classification

We evaluate the proposed positional encoding on image classification, another popular task on images, based on Vision Transformer (ViT) [9], a Transformer-only architecture that does not use CNN for image embedding. The default positional encoding in ViT is Embed-1D. In this experiment, we focus on the ViT-B/16 model that is a 12-layer Transformer encoder with Hidden_size=768, MLP_size=3072 and 12 attention heads. The input to ViT-B/16 uses a 14×14 image grid where each cell corresponds to a 16×16 image patch. We train each model on the ImageNet dataset for 90 epochs, and report its accuracy on the ImageNet validation dataset. Learnable Fourier+MLP achieved better performance (Precision@1=74.5%) on the validation dataset than Embed1D (Precision@1=73.6%).

Dosovitskiy et.al. investigated several positional encoding methods in their work (see Table 8 in [9], including Embed-1D, Embed-2D and relative positional encoding. They pre-trained these models on the large JFT dataset (300 million examples) and then report their performance on ImageNet 5-shot linear tasks. They found the model suffers when no positional encoding is used but there are no significant impacts for using each of these positional encoding methods. We suspect that given such a large model (86M Params) and the small number of unique positions ($14 \times 14 = 196$) on the image, it is not difficult for any of these positional encoding methods to learn, and the simple Embed1D performs competently that achieved 64.206% accuracy. We experimented Learnable-Fourier+MLP in this experiment, which achieved 64.732% accuracy.

4.4 Widget Captioning

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 [20], 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*. Li et al. found that position encoding has a significant impact on the performance of widget captioning models (see Table 6 in Appendix F of the paper [20]).

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, i.e., $G = 1$; Learnable-Fourier+MLP-2/2 splits the 4 coordinates into 2 groups `[(top, left), (bottom, right)]`, i.e., $G = 2$; and finally Learnable-Fourier+MLP-4/1 encodes 4 groups of 1 coordinate value `[(top), (left), (bottom), (right)]`, i.e., $G = 4$. 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*).

Table 3: The performance 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.

Positional Embedding	BLEU-1	BLEU-2	ROUGE	CIDEr	METOER	SPICE
SOTA [20]	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

We use the same model architecture and hyperparameters of the strongest model, *Pixel+Local+Context*, as the original paper [20], and built our experiment based on the public codebase of widget captioning[5]. 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 3, 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-Fourier + MLP 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 ($p > 0.05$). 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 treat 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

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

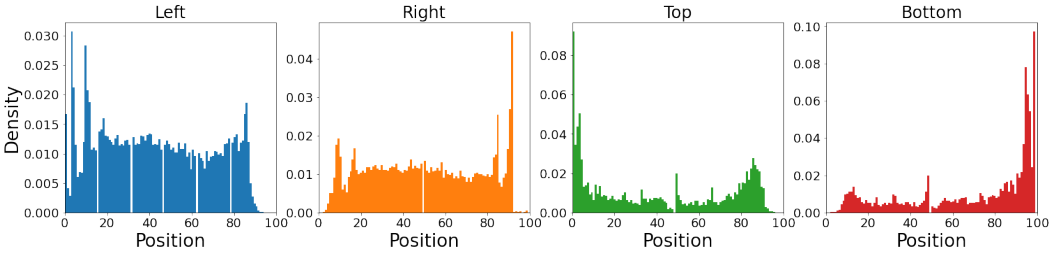


Figure 4: Widget position distributions for each dimension in the training set. There are positions rarely occurred in the training set.

perceptron (Equation 6) 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.

Table 4: The accuracy for widgets with seen and unseen positions.

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**

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 4). There are 1867 widgets with Seen positions and 2692 with unseen positions from the test set. Table 4 shows that our method generalizes to unseen positions significantly better than baselines. There are a number of reasons for the proposed positional encoding to generalize for unseen positions. First, it treats positions as continuous values. As a result, it does not suffer from the difficulty with embedding-based approaches where an embedding vector is assigned to a discrete position, which can be not trained or significantly under-trained when a position is unseen or rarely seen. Second, the Fourier features capture the relative positional relationships by maintaining the shift-invariant property during learning (Equation 3), which applies to unseen positions as well.

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. Our work investigated positional encoding when it is combined with content embedding via addition. It would be interesting to investigate how our positional encoding performs when it is concatenated with content embedding.

Although Euclidean distances might be a desirable positional metric for images, tasks such as widget captioning involves sparse spatial structures, and the spatial relations between two rectangular objects on the screen can be more complicated. For example, similarity between two bounding boxes can be related to their vertical or horizontal alignment or overlaps (IoU), or other domain specific factors related to UI layouts. Our positional encoding method outperformed benchmark methods in this task, which showed that it is better equipped to capture these spatial relationships. Yet, it is worth investigating methods that can more directly capture these complex spatial relationships.

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, image classification, and sparse spatial structure modeling in user interfaces, which show that our positional encoding consistently outperforms the benchmark methods.

Acknowledgments

We would like to thank anonymous reviewers for their thoughtful comments and constructive feedback that have significantly improved the work.

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. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 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] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Do we really need explicit position encodings for vision transformers? *CoRR*, abs/2102.10882, 2021.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *ICCV 2021*, abs/2104.01136, 2021.

[12] 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.

[13] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

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

[15] 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.

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

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

[18] 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.

[19] 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*, pages 8198–8210, Online, July 2020. Association for Computational Linguistics.

[20] 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.

[21] 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.

[22] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

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

[24] Antoine Liutkus, Ondřej Cífka, Shih-Lun Wu, Umut Simsekli, Yi-Hsuan Yang, and Gael Richard. Relative positional encoding for transformers with linear complexity. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7067–7079. PMLR, 18–24 Jul 2021.

[25] 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.

[26] 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.

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

[28] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2017.

[29] 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.

[30] 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.

[31] 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.

[32] 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. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[33] 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.

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

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

[36] 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.

[37] 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.

[38] 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.

[39] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in {bert}. In *International Conference on Learning Representations*, 2021.

[40] Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[41] 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.

[42] 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.

[43] 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. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc., 2019.

A Attention-Based Models

We review positional encoding in the context of Transformer models [38]. The central building block of these models is multi-head attention and each attention head is calculated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{7}$$

where queries $Q \in \mathcal{R}^{N \times d_k}$, keys $K \in \mathcal{R}^{N \times d_k}$, and values $V \in \mathcal{R}^{N \times D_v}$. N is the number of items to consider, e.g., the number of tokens in a sequence or the number of pixel patches in an image. d_k is the dimension of a key and query, and D_v is the dimension of a value vector. Queries, keys and values are acquired via a linear projection of the input at each attention layer. For self-attention, they share the same input:

$$Q = E_X M_Q; K = E_X M_K; V = E_X M_V \tag{8}$$

where $M_Q \in \mathcal{R}^{|E_X| \times d_k}$, $M_K \in \mathcal{R}^{|E_X| \times d_k}$ and $M_V \in \mathcal{R}^{|E_X| \times d_v}$ are the linear projection. $E_X \in \mathcal{R}^{N \times |E_X|}$ is the embedding of input X, which is jointly represented by its content embedding, C_X, and its positional encoding, P_X.

$$E_X = C_X \oplus P_X \tag{9}$$

where $\oplus$ can be either concatenation or element-wise addition. Previous work has investigated different combinations and decomposition of positional encoding and content embedding [13]. While concatenation and addition provide comparable results, the lack of positional encoding, P_X, will cause a significant drop in accuracy [38, 3]. In this paper, we investigate methods for realizing P_X.

B Hyperparameters & Parameter Sizes

Positional Embedding	Reformer Model Parameter Size
Embed-1D	73.2M
Embed-2D	60.7M
Sine-1D	60.6M
Sine-2D	60.6M
Transformer MD-Sine	60.6M
Transformer MD-Sine + MLP	60.7M
MLP	60.6M
MLP + MLP	60.7M
Fourier	60.6M
Fourier + MLP	60.7M
Learnable-Fourier	60.6M
Learnable-Fourier + MLP	60.7M

Table 5: The model parameter sizes of Reformer [16] with different positional encoding methods.

For Reformer experiments, each model is based on the Reformer model for the Imagenet64 task [16]. The number of parameters for each Reformer model is summarized in Table 5. We here focus on the positional encoding part of the model that is where each approach differs. Our positional encoding uses roughly the same number of trainable parameters as Embed-2D, the benchmark method used in the original Reformer. For all the Fourier-based methods, including Learnable-Fourier+MLP, we used $|F| = 768$, $|H| = 32$, $D = 768$ and $\gamma = 1.0$. For the MLP modulators, we used LayerNorm before the dense projections, W_1 and W_2 (Algorithm 1 Line 2), and a dropout rate of 10% after ReLu, the non-linear activation. We set $G = 1$ because vertical and horizontal positions need to be

Positional Embedding	DETR Model Parameter Size
Embed-2D	41.6M
Normalized Sine-2D	41.6M
Unnormalized Sine-2D	41.6M
MLP	41.6M
Learnable-Fourier + MLP	41.6M

Table 6: The model parameter sizes of DETR [3] with different positional encoding methods.

Positional Embedding	Model Parameter Size
SOTA [20]	5.11M
Embed-4D	5.11M
Sine-4D	5.07M
Learnable-Fourier-2/2	5.07M
Fixed-Fourier+MLP-1/4	5.10M
Fixed-Fourier+MLP-2/2	5.08M
Fixed-Fourier+MLP-4/1	5.07M
Learnable-Fourier+MLP-1/4	5.11M
Learnable-Fourier+MLP-2/2	5.07M
Learnable-Fourier+MLP-4/1	5.07M

Table 7: The model parameter sizes of Widget captioning model [20] with different positional encoding methods.

mapped jointly to model the inductive bias of L2 distances on an image. Embed-1D uses significantly more parameters because it needs to assign an embedding vector for each position in a flattened image. Sine-1D and Sine-2D are parameter-free encoding, thus use the least parameters. 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[6]. The training for each Reformer model is parallelized across 32 TPU v2 cores, and each batch contains 8 sequences (images) on each core. We trained each model variant for 100k steps, which took about 24 hours to complete.

The parameter sizes for each DETR model [3] are shown in Table 6. All the variants of DETR roughly uses the same number of trainable parameters. We used $\gamma = 1.0$ for Learnable-Fourier + MLP. The MLP uses a dense layer 2×256 with ReLu as activation, which is then followed by another dense layer of 256×256. We did not use any dropout in these methods. 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. All the variants are based on the DETR codebase[7], which are ported into JAX[8], 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. We trained each variant for 90k steps, which took roughly 16 hours to complete.

For UI Widget Captioning experiments, the number of parameters of each model variant is shown in Table 7. The model architecture that is shared by each model variant is summarized in the paper and detailed in the previous paper [20]. For Fourier-based methods, we used $|F| = 128, 64, 32$, $G = 1, 2, 4$ for position grouping variants: 1/4, 2/2 and 4/1, respectively. We used $\gamma = 100$ for initializing W_r for all the Fourier-based methods. We used a dropout of 20% after the non-linear activation in the MLP modulator. We use the same model architecture and hyperparameters of the strongest model, *Pixel+Local+Context*, as the original paper [20], and built our experiment based on the public codebase of widget captioning[9]. 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 NVIDIA V100 GPU.

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

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

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

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

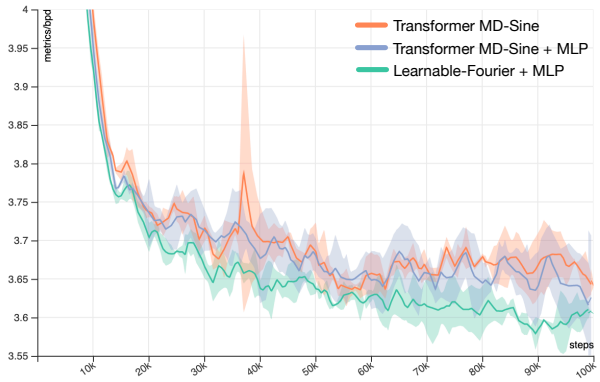


Figure 5: Bits per dim (bpd) w.r.t. training steps on the image generation task with Reformer. The ablation compares learnable Fourier features with multi-dimensional sinusoidal PE based on Transformer frequencies. The plot shows the mean and 95% confidence interval based on 3 repeats of experiments for each method.

For computational complexity, embedding-based approaches generally require less computation compared to others. The Reformer's default PE, Embed2D, trains at 1.86 steps/second. Sine2D in contrast trains at 1.81 steps/second, and our Learnable-Fourier+MLP is slower and trains at 1.22 steps/second. For the models for the object detection and widget captioning tasks, the impact of using different PE on runtime is negligible because the rest computation in the training is more dominant, e.g., Hungraian matching for computing the minimal loss in DETR for object detection.

C Additional Ablation Studies

It is possible to extend traditional Sinusoidal positional encoding (Equation 1) for the multi-dimensional positions without using concatenation. We conduct the ablation for 2D positions by linearly combining the vertical and horizontal positions using constant frequencies, which these frequencies are manually determined. In this ablation, we adapt the original Transformer Sinusoidal frequencies for each dimension. Specifically, for a 2D position (x, y), the multi-dimensional Sinusoidal PE, referred as *Transformer MD-Sine*, is the follow, where D is the dimension of the PE and $0 \leq d \leq \frac{D}{2}$.

$$PE(p, 2d) = \sin\left(\frac{x}{10000^{2d/D}} + \frac{y}{5000^{2d/D}}\right); PE(p, 2d+1) = \cos\left(\frac{x}{10000^{2d/D}} + \frac{y}{5000^{2d/D}}\right)$$

As shown in Figure 5, Transformer MD-Sine performs poorly. Although adding MLP improves its performance, it still does not perform as good as Learnable Fourier. Although it is possible to find better frequencies for each dimension, it can be effort consuming to manually tune these frequencies to perform optimally.

We added additional MLP layers for

D Learned Positional Encoding Analysis

Our positional encoding is seeded with Fourier features whose dot product approximates L2 distances—that brings the inductive bias to the model, which then evolves as learning progresses. In this section, we analyze the positional encodings learned from the Reformer and Widget Captioning tasks. Note that the following analysis is focused on the output of that of Equation 2 instead of Equation 6. This is because the Fourier features directly represent the position while the MLP is trained to modulate the positional encoding to merge with the content embedding. It is less informative to analyze the MLP output because it does not directly participate in the dot product attention (Equation 7). In all the image benchmarks, the MLP output will be added to the content embedding and the addition is further processed by the transformation with M_k in Equation 8. in the widget

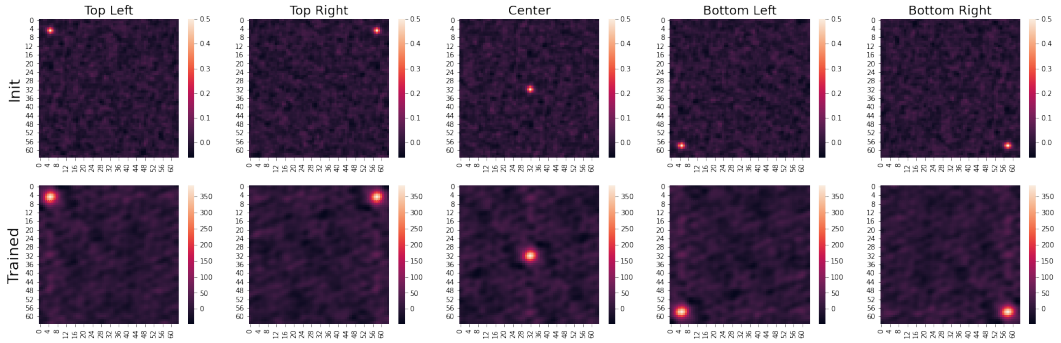


Figure 6: The positional similarity, $r_x \cdot r_y$, of different positions on an image, to the rest of the positions on an image, as learned by Learnable-Fourier+MLP without using the MLP modulator in Reformer. The Fourier features are initialized with weights drawn from a Gaussian distribution: $\gamma = 1.0$. The Top-Left, Top-Right, Center, Button-Left, and Bottom-Right positions are at (4, 4), (4, 57), (31, 31), (57, 4), (57, 57) in the coordinates.

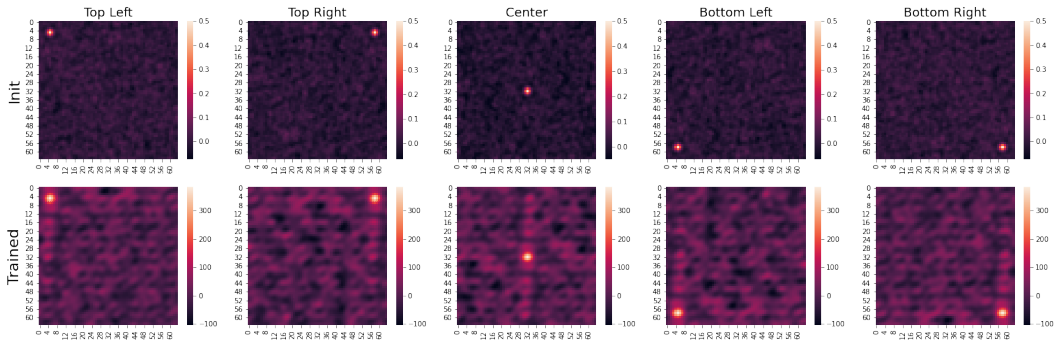


Figure 7: The positional similarity, $r_x \cdot r_y$, of different positions on an image, to the rest of the positions on an image, as learned by Learnable-Fourier without using the MLP modulator in Reformer. The Fourier features are initialized with weights drawn from a Gaussian distribution: $\gamma = 1.0$.

captioning benchmark, the MLP output will be concatenated with the content embedding and then transformed by a dense layer before further transformed by M_k.

D.1 PE Analysis for Imagenet64 Reformer Tasks

Figure 6 visualizes the similarity of a given position on a 64×64 image, to the rest of the positions on the image, at the initial stage and the end of the training. The similarity is computed based on the dot product of the positional encoding of these positions. The first row, Init, shows the similarity heatmap resulted from the initially seeded Fourier features based on $\gamma = 1.0$. The second row, Trained, shows the similarity from the positional encoding trained after 100K steps when the model converges. As we can see, the positional relationship becomes less concentrated than the initialization, i.e., the "ball" becomes larger. To further understand the impact of having the MLP modulator on the positional encoding, we compare the learned positional encoding with and without the MLP modulator. When there is no MLP modulator (Figure 7, the learned positional encoding is less clean than the one with MLP. We suspect it is because without MLP, the positional encoding needs to directly participate in the addition with the content embedding (Equation 9). As a result, the encoding is not only learning to represent positions but also pressured to work with content embedding. As we show in our experiments, the lack of the MLP modulator often results in a decrease in accuracy.

D.2 PE Analysis for Widget Captioning Tasks

Positional relationships are more complex in the Widget Captioning task, because each position is defined as a four-coordinate bounding box. We consider point-wise similarity a building block

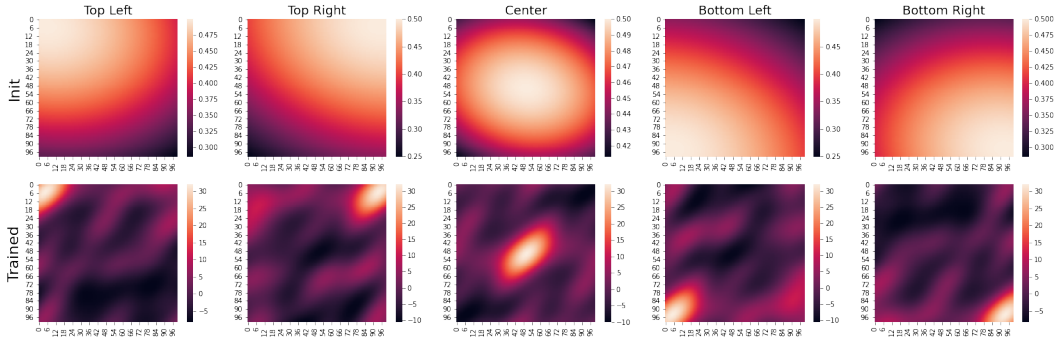


Figure 8: The positional similarity, $r_x \cdot r_y$, of different positions, to the rest positions on a UI screen, learned by Learnable-Fourier+MLP-2/2 in Widget Captioning. Note that in this task, each position is defined as a 4-coordinate bounding box. The heatmap only visualizes the point-wise similarity. The Fourier features are initialized with weights drawn from a Gaussian distribution: $\gamma = 64.0$.

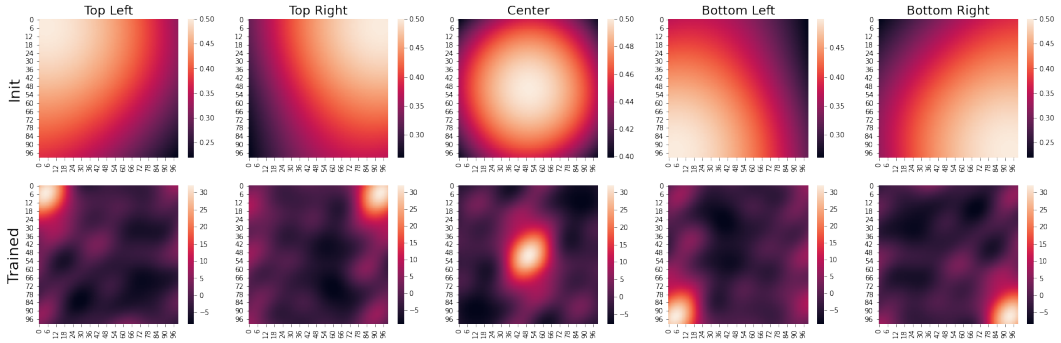


Figure 9: The positional similarity, $r_x \cdot r_y$, of different positions, to the rest positions on a UI screen, learned by Learnable-Fourier+MLP-2/2 with the KL loss (Equation 10 and 11) in Widget Captioning. The Fourier features are initialized with weights drawn from a Gaussian distribution: $\gamma = 1.0$.

for bounding box similarity as discussed in the paper. Figure 8 shows the point-wise positional similarity learned by Learned-Fourier+MLP 2/2, which groups four coordinates into two groups to represent the top-left corner and the right-bottom corner of a bounding box. In this task, we see a more spread positional relationship than that of the Imagenet64 Reformer task, because we seed the Fourier features with $\gamma = 100$. We observed that the positional relation becomes more concentrated over the course of the training than that of the initial encodings. We also see the positional relation distribution becomes more skewed (towards the anti-diagonal direction). To understand whether maintain the symmetry of the distribution would help on accuracy, we conduct additional experiments by applying a regularizer to the Fourier weights W_r (Algorithm 1 Line 1) as the follow.

$$\mathcal{L}_{KL} = -\frac{1}{2}(1 - \log \bar{\sigma}^2 + \log \sigma^2 - \frac{\sigma^2 + \mu^2}{\bar{\sigma}^2}) \tag{10}$$

where μ and σ^2 are the mean and variance of W_r. $\bar{\sigma}^2$ is the target variance that is also learnable, which is initialized as γ^{-2}. The KL loss ensures W_r to obey a Gaussian distribution centered at 0 thus maintains the symmetry of positional relationships along all the directions. When training the model, the regularizer loss $\mathcal{L}_{KL}$ is added to the overall loss for optimization.

$$\mathcal{L}_{total} = \mathcal{L}_{model} + \alpha \mathcal{L}_{KL} \tag{11}$$

In this experiment, we use $\alpha = 1$. The resulted positional encoding is shown in Figure 9. As we can see, the symmetry of the positional relation distribution is better maintained with the KL loss. We see a clear improvement of accuracy with the use of this KL loss forLearned-Fourier+MLP 2/2. However, using the KL loss does not seem to help image-based tasks much (Imagenet64 and DETR tasks).

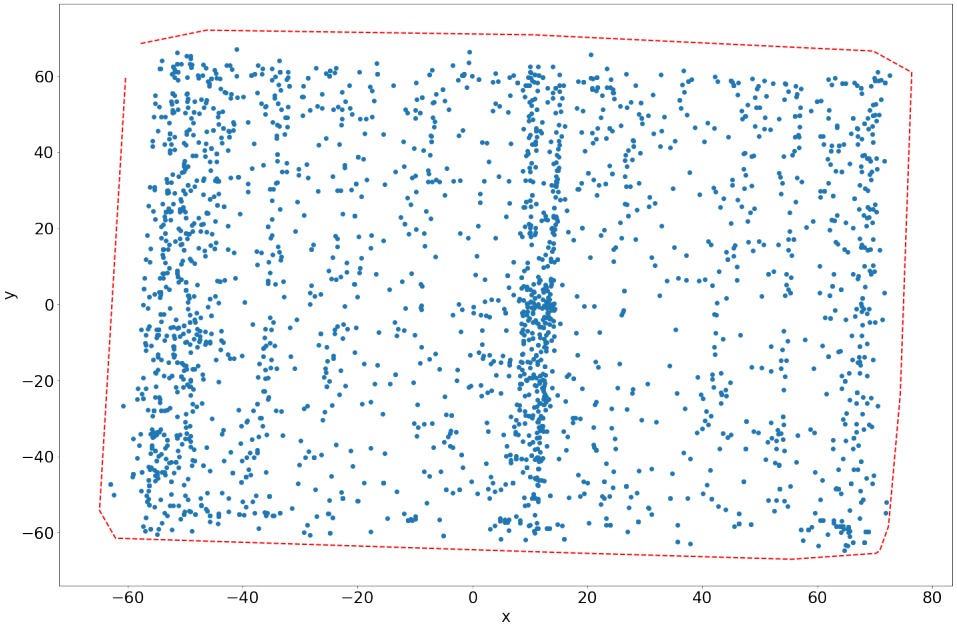


Figure 10: Unseen positions in test set with convex hull of all the positions in training set.

We suspect that as shown in Figure 6, the symmetry of positional relation distribution is naturally maintained even without using the KL loss. Thus KL loss is less useful in such cases.

E Unseen Position Distribution in Widget Captioning Dataset

To show the distribution of the unseen positions in test set, we map all these unseen positions to 2D with PCA and plot them in the Figure 10 below. We also plot the 2D convex hull of all the positions in the training set (mapped via PCA) as the red dashed line. We can see all the unseen positions are within the convex hull of the training positions.