RelFlexformer: Efficient Attention 3D-Transformers for Integrable Relative Positional Encodings

Byeongchan Kim[*1] **Arijit Sehanobish**[*2] **Avinava Dubey**[*3]
Min-hwan Oh[1] **Krzysztof Choromanski**[*4,5]

[1]Seoul National University [2]Kensho Technologies [3]Google Research
[4]Google DeepMind [5]Columbia University

Abstract

We present a new class of efficient attention mechanisms applying universal 3D Relative Positional Encoding (RPE) methods given by arbitrary integrable modulation functions f. They lead to the new class of 3D-Transformer models, called *RelFlexformers*, flexibly integrating those RPEs, and characterized by the $O(L \log L)$ time complexity of the attention computation for the L-length input sequences. RelFlexformers builds on the theory of the Non-Uniform Fourier Transform (NU-FFT), naturally generalizing several existing efficient RPE-attention methods from structured settings with tokens homogeneously embedded in unweighted grids into general non-structured heterogeneous scenarios, where tokens' positions are arbitrarily distributed in the corresponding 3D spaces. As such, RelFlexformers can be applied in particular to model point clouds. Our extensive empirical evaluation on a large portfolio of 3D datasets confirms quality improvements provided by the NU-FFT-driven attention modulation techniques in the RelFlexformers.

1 Introduction

Attention-based Transformers [75] have become dominant architectures for 3D learning, but their standard form is not aligned well with point cloud (PC) data modality and other geometric data. In these settings, the input tokens are unordered, irregularly spaced in Euclidean space, and often carry strong inductive bias through relative positions [48]. This has motivated a large family of relative positional encoding (RPE) methods, which inject geometry into attention and improve performance in 3D classification and segmentation [93]. At the same time, the quadratic cost of exact attention remains a major bottleneck, especially for large point sets and dense geometric models.

Efficient attention mechanisms, such as Performers [12], offer a compelling alternative by replacing the softmax kernel with linear kernels but in the nonlinearly transformed domains, reducing the cost of attention from quadratic to linear time. However, integrating geometric modulation into such models is not straightforward. Unlike standard attention, Performers do not materialize the attention matrix explicitly, so a geometric mask cannot simply be inserted as an element-wise correction. Any viable modulation must instead be compatible with the kernelized formulation and admit fast matrix-vector products [11]. This requirement is essential: without it, the geometric bias would destroy the computational advantage that makes the architecture useful in the first place.

In this work, we introduce *RelFlexformers*, a class of efficient 3D Transformers that incorporate a broad family of RPEs through an arbitrary integrable modulation function f. Our key observation is that the resulting masked attention operator can still be applied efficiently, provided the modulation

*Equal Contribution. Project page: `https://relflexformer.github.io/`
Correspondence to: bckim97@snu.ac.kr, arijit.sehanobish1@gmail.com, kchoro@google.com.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

admits a fast subquadratic matrix-vector multiplication. We provide such an implementation and show that it can be realized in $O(L \log L)$ time for L-length sequences, using a *Non-Uniform Fast Fourier Transform* (NU-FFT) formulation. This allows geometric information to be incorporated directly into Performer-style attention without reverting to explicit $L \times L$ attention computation.

RelFlexformers generalize several existing efficient RPE attention methods, typically designed for structured token layouts such as regular grids or homogeneous spatial lattices. In contrast, our formulation applies to arbitrary 3D token configurations, making it well suited to point clouds, depth images (with tokens embedded into 3D spaces, for instance via lifting with the average depth signal across pixels in the patch) and other irregular geometric inputs. The resulting framework is flexible: different modulation functions induce different geometric priors, while the underlying attention computation remains efficient.

We evaluate RelFlexformers on a range of 3D point cloud benchmarks for classification and segmentation. Across these tasks, the proposed NU-FFT-driven modulation consistently improves performance over strong Performer-based baselines and yields a practical route for combining geometric expressivity with subquadratic attention. RelFlexformer is closing the accuracy gap, as compared to the regular quadratic Transformer's attention, often outperforming it.

Our main contributions are:

1. We formulate a general class of 3D RPE-modulated efficient attention mechanisms based on arbitrary L_1-integrable functions. We derive an $O(L \log L)$ implementation via NU-FFT, enabling fast masked attention without materializing the attention matrix (Sec. 3).
2. We validate RelFlexformer as a versatile, drop-in replacement across diverse backbones (PCT, PTv3, DFormer), consistently elevating baseline Performer attention across synthetic models, sparse LiDAR, and multi-sensor RGB-D data. We also show that RelFlexformer not only bridges the performance gap inherent to efficient linear attention, but actually surpasses standard $O(L^2)$ Transformers on various datasets. Moreover, we show that our distance-based RPE modulation is orthogonal and highly complementary to token-level encodings like PointRoPE and can be combined together (Sec. 4).

2 Related Work

Efficient Transformers and Kernel Attention. Standard Transformers scale quadratically, $O(L^2)$, with sequence length L [75]. To mitigate this, one line of research focuses on sparse or local attention [31, 4, 102]. Another approach uses kernel-based linear attention to rewrite the softmax operator, achieving $O(L)$ complexity [30]. Specifically, Performers [12] replace softmax kernels with linear (dot-product) kernels, but in new spaces, obtained via nonlinear transformations of queries and keys. Those mechanisms include in particular techniques applying randomized maps (via random features) to unbiasedly approximate original softmax kernel. While efficient, these kernelized methods often struggle to integrate structured biases like RPEs without reverting to quadratic cost.

3D Point Cloud Transformers. Transformers are naturally suited for 3D point clouds due to their permutation equivariance. Architectures like PCTs [23] and Point Transformers [107] (and their successors) have achieved state-of-the-art results by applying self-attention to local neighborhoods or global point sets. However, these models typically rely on $O(L^2)$ attention or local grouping heuristics to handle geometric relationships. While some models incorporate RPEs via Euclidean distances, they do not provide a mechanism for global, sub-quadratic attention that maintains a general integrable RPE mask. Separately, Point Transformer V3 (PTv3) [84] improves scalability by serializing point clouds with space-filling curves and applying dot-product attention within serialized patches. By replacing KNN-based neighborhood construction and explicit RPE with serialized neighbor mapping and sparse-convolutional conditional positional encoding, PTv3 achieves a much larger receptive field with improved speed and memory efficiency. Nevertheless, its efficiency comes from restructuring point clouds into serialization-induced local neighborhoods, rather than from a global sub-quadratic attention mechanism with a general integrable RPE mask.

Relative Positional Encodings (RPE). RPE enhances Transformers by encoding the spatial relationship between tokens [66]. Recent advances include ALiBi [55], RoPE [69] and Fourier Positional Encoding [28], which improve extrapolation. However, most RPE schemes assume a 1D sequence or

2D and 3D grid structures [63]. In 3D point clouds, the RPE should ideally depend on the continuous distance $f(\mathbf{x}_i - \mathbf{x}_j)$. Luo et al. [49] demonstrated that 1D RPE can be computed in $O(L \log L)$ via FFT by exploiting Toeplitz structures. [11] provides the first framework to incorporate arbitrary RPE masks in the Performer-style attention, provided the *mask matrix supports fast (sub-quadratic) matrix vector multiplication*. This idea was utilized in [13], where relative positional encodings (RPE) are implemented using a tree-based mask. The original graph is approximated by a spanning tree, allowing the mask to be applied efficiently via fast matrix–vector multiplication, and thereby integrated into the Performer architecture. [60] uses the graph random features (GRF) to provide inductive biases to linear transformers. These approaches rely on an auxiliary structure, such as a regular grid or a graph/tree, and require additional preprocessing to obtain efficient mask application. In contrast, our goal is to construct the geometric mask itself implicitly for arbitrary non-equispaced 3D coordinates, without materializing a mask or constructing a KNN graph or tree. Our work generalizes the Toeplitz based masks on structured grids and runs in subquadratic time. Since point clouds are non-equispaced, we utilize Non-Uniform Fast Fourier Transform (NU-FFT) [22, 17] to enable arbitrary 3D RPE masks within a Performer framework, achieving $O(L \log L)$ complexity for irregular geometries. Crucially, this formulation provides a unified geometric operator that subsumes recent multidimensional encodings like STRING [63] as special cases, while extending the same sub-quadratic efficiency to dense RGB-D data via coordinate lifting. We provide a detailed discussion on the evolution of RGB-D representations and their intersection with our unified RPE framework in App. A.

3 RelFlexformer

The RelFlexformer attention mechanism relies on kernelized attention via postive random features [12] and the theory of the Non-Uniform Fast Fourier Transform (NU-FFT). We provide a basic introduction to Fourier analysis in Sec. 3.1 and show how it is applied in RelFlexformer in Sec. 3.2.

3.1 Preliminaries

The Fourier Transform (FT) and Inverse Fourier Transform (IFT) of the L_1-integrable functions $f, g : \mathbb{R}^r \to \mathbb{R}$ are defined as follows, for the imaginary unit $i \in \mathbb{C}$ ($i^2 = -1$):

$$\mathcal{F}_f(\xi) = \int_{\mathbb{R}^r} \exp(-2\pi i \mathbf{x}^\top \xi) f(\mathbf{x}) d\mathbf{x}, \; \mathcal{F}_g^{-1}(\mathbf{x}) = \int_{\mathbb{R}^r} \exp(2\pi i \mathbf{x}^\top \xi) g(\xi) d\xi \tag{1}$$

The *Fourier Inversion Theorem* [19] states that taking the Inverse Fourier Transform of the Fourier Transform of a given function f, or vice versa, results in the function f itself:

$$\mathcal{F}_{\mathcal{F}f}^{-1} = \mathcal{F}_{\mathcal{F}^{-1}f} = f. \tag{2}$$

Theorem 3.1 (Convolution Theorem [19])**.** *Take two functions $g, h : \mathbb{R}^r \to \mathbb{R}$ and their convolution:*

$$p(\mathbf{z}) = (g \star h)(\mathbf{z}) \stackrel{\text{def}}{=} \int_{\mathbb{R}^r} g(\tau) h(\mathbf{z} - \tau) d\tau \tag{3}$$

Then the following holds:

$$\mathcal{F}_p = \mathcal{F}_g \mathcal{F}_h; \; \mathcal{F}_p^{-1} = \mathcal{F}_g^{-1} \mathcal{F}_h^{-1} \tag{4}$$

The FT or IFTs of the convolution of two functions is the product of the FTs or IFTs of those functions. Thus convolving in the original space corresponds to the multiplication in the Fourier space. Fourier analysis can be extended to objects that are not valid functions, but the so-called *distributions*. One of the most standard examples is a Dirac delta function $\delta_{\mathbf{z}_0}$, defined as follows for $\mathbf{z} \in \mathbb{R}^r$:

$$\delta_{\mathbf{z}_0}(\mathbf{z}) = \begin{cases} 0 & \text{if } \mathbf{z} \neq \mathbf{z}_0 \\ +\infty & \text{if } \mathbf{z} = \mathbf{z}_0 \end{cases} \tag{5}$$

It can be shown that $\mathcal{F}_{\delta_{\mathbf{z}_0}}(\xi) = \exp(-2\pi i \xi^\top \mathbf{z}_0)$ and $\mathcal{F}_{\delta_{\mathbf{z}_0}}^{-1}(\xi) = \exp(2\pi i \xi^\top \mathbf{z}_0)$.

Definition 3.2 (Non-Uniform Fast Fourier Transform (NU-FFT))**.** For $a, b \in \mathbb{N}$, two sequences of vectors in $\mathbb{R}^d$: $(\alpha_1, ..., \alpha_a), (\beta_1, ..., \beta_b)$ and a sequence of scalars $(c_1, ..., c_b)$, the Non-Uniform Fast Fourier Transform is a transformation computing a sequence of scalars: $(\gamma_1, ..., \gamma_a)$ defined as:

$$\gamma_k = \sum_{j=1}^{b} c_j \exp(2\pi i \alpha_k^\top \beta_j) \tag{6}$$

The Non-Uniform FFT can be computed in time $O((a+b)\log(a+b))$.

Remark : The above calculations boil down to the multidimensional Discrete Fourier Transform, or DFT (for $d=1$, the usual DFT) when the vectors α and β form a Cartesian grid, i.e. β_j must be of the form $\left(\frac{j_1}{N_1}, \frac{j_2}{N_2}, \ldots, \frac{j_d}{N_d}\right)$, where each j_i is an integer, and α_k are integer vectors $(k_1, k_2, \ldots, k_d)$. Those can be computed efficiently with the use of the standard Fast Fourier Transform (FFT; [19]).

3.2 NU-FFT for RPE-modulated attention

In this section, we introduce our RPE-attention mechanism, starting from the definition of masked attention. The mask encodes how the RPE mechanism is modulating regular attention computations.

Denote by $\{\mathbf{x}_i\}_{i=1}^L \subseteq \mathbb{R}^d$ the embeddings of input tokens to the attention mechanism and by $\mathbf{X} \in \mathbb{R}^{L\times d}$ a matrix of rows given by $\{\mathbf{x}_i\}_{i=1}^L$. We define by $\{\mathbf{q}_i\}_{i=1}^L, \{\mathbf{k}_i\}_{i=1}^L \subseteq \mathbb{R}^{d_{QK}}, \{\mathbf{v}_i\}_{i=1}^L \subseteq \mathbb{R}^d$ the sets of queries, keys and values respectively, obtained by applying three learnable linear transformations: $\mathbf{W}_Q \in \mathbb{R}^{d_{QK}\times d}, \mathbf{W}_K^{d_{QK}\times d}, \mathbf{W}_V \in \mathbb{R}^{d\times d}$ to embeddings $\{\mathbf{x}_i\}_{i=1}^L$.

Algorithm 1: $\mathrm{FastMult}_{\mathbf{M}}$**: NU-FFT Mask Multiplication**

Input: Input vector $\mathbf{u} \in \mathbb{R}^L$, token coordinates $\{\mathbf{r}_i\}_{i=1}^L \in \mathbb{R}^d$, spatial modulation function f, quadrature samples $\{\xi_s\}_{s=1}^S$ and coefficients $\{a_s\}_{s=1}^S$.

Output: Vector $\mathbf{w} \in \mathbb{R}^L$ approximating $\mathbf{Mu}$.

1. Compute the Fourier Transform of the point cloud signal at sampled frequencies $s \in \{1,\ldots,S\}$ using the forward Non-Uniform FFT in $O(L\log L)$ time:
   $\mathcal{F}_P(\xi_s) = \sum_{l=1}^L u_l \exp(-2\pi i \xi_s^\top \mathbf{r}_l)$.
2. Calculate the modulated coefficients for each frequency sample using the spatial mask's Fourier Transform:
   $b_s = a_s \mathcal{F}_f(\xi_s)\mathcal{F}_P(\xi_s)$.
3. Output the final evaluated function at the point coordinates $i \in \{1,\ldots,L\}$ using the inverse Non-Uniform FFT in $O(L\log L)$ time:
   $\mathbf{w}_i = \sum_{s=1}^S b_s \exp(2\pi i \xi_s^\top \mathbf{r}_i)$.

The unnormalized attention matrix $\mathbf{A} \in \mathbb{R}^{L\times L}$ is given as: $\mathbf{A} = [\exp(\mathbf{q}_i^\top \mathbf{k}_j)]_{i,j\in\{1,\ldots,L\}} \in \mathbb{R}^{L\times L}$. In the regular Relative Positional Encoding (RPE) mechanism, this matrix is then modulated by the so-called *mask matrix* $\mathbf{M} = [f(\mathbf{r}_i - \mathbf{r}_j)]_{i,j\in\{1,\ldots,L\}} \in \mathbb{R}^{L\times L}$, where the set $\{\mathbf{r}_i\}_{i=1}^L \in \mathbb{R}^d$ encodes tokens' positions in some geometric space (e.g. 3D-space for point cloud applications), f is called the *modulation function* and the modulation itself is given via a Hadamard (element-wise) product: $\overline{\mathbf{A}} = \mathbf{A} \odot \mathbf{M}$. Matrix $\overline{\mathbf{A}}$ is then row-wise normalized and the resulting matrix $\overline{\mathbf{A}}_{\text{norm}} = [\frac{\overline{\mathbf{A}}_{i,j}}{\sum_{k=1}^K \overline{\mathbf{A}}_{i,k}}]$ is applied to the value matrix $\mathbf{V} \in \mathbb{R}^{L\times d_V}$, of rows given by $\{\mathbf{v}_i\}_{i=1}^L$. The final result of attention is given as follows, where the columns of the output matrix are interpreted as new embeddings:

$$\mathbf{X} \rightarrow \mathbf{X} + \overline{\mathbf{A}}_{\text{norm}}\mathbf{V} \tag{7}$$

Note that the brute-force application of that mechanism requires explicit materialization of the mask matrix $\mathbf{M} \in \mathbb{R}^{L\times L}$, which already leads to quadratic time $O(L^2)$. To bypass this bottleneck, we build upon linear-attention frameworks such as Performers [12]. Specifically, regular softmax attention $\mathbf{A}_{i,j} = \exp(\mathbf{q}_i^\top \mathbf{k}_j)$ is approximated via a lower-rank factorized counterpart $\tilde{\mathbf{A}}_{i,j} = \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j)$ using a positive random feature map $\phi : \mathbb{R}^{d_{QK}} \rightarrow \mathbb{R}_+^m$ (e.g., $\phi(\mathbf{x}) = \frac{1}{\sqrt{m}} \exp\left(-\frac{\|\mathbf{x}\|^2}{2}\right) \exp(\boldsymbol{\omega}^\top \mathbf{x})$ with Gaussian random samples $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$). The key advantage of this factorization is that it avoids explicitly materializing the $L \times L$ attention matrix.Instead, by associativity of matrix multiplication, the keys and values can first be aggregated as $\sum_{j=1}^L \phi(\mathbf{k}_j)\mathbf{v}_j^\top$, after which each query feature $\phi(\mathbf{q}_i)$ is applied to this shared $m \times d_V$ representation. Thus, for fixed feature dimension m, the resulting computation scales linearly with L rather than quadratically.

The positive random feature approximation removes the quadratic cost associated with the unmodulated attention matrix. To incorporate RPEs without reintroducing this quadratic cost, we follow the approach of [11], which reduces the computation of RPE-modulated linear attention to matrix-vector products with the mask $\mathbf{M}$. Specifically, the resulting complexity is $O(L + T(\mathbf{M}))$, where $T(\mathbf{M})$ denotes the cost of computing $\mathbf{Mu}$ for an arbitrary $\mathbf{u} \in \mathbb{R}^L$. Importantly, this formulation does not require explicitly constructing the $L \times L$ mask $\mathbf{M}$. For the relative-position mask $\mathbf{M}_{ij} = f(\mathbf{r}_i - \mathbf{r}_j)$, we show that its action on an arbitrary vector can be approximated directly in $O(L \log L)$ time using the NU-FFT. Thus, the central computational problem is to evaluate $\mathbf{Mu}$ efficiently, without ever materializing $\mathbf{M}$.

Take an arbitrary $\mathbf{u} \in \mathbb{R}^L$ and $\mathbf{w} = \mathbf{Mu}$. Note that we can re-write $\mathbf{w}_i$ for $i = 1, ..., L$ as:

$$\mathbf{w}_i = w(\mathbf{r}_i), \text{where } w(\mathbf{z}) \stackrel{\text{def}}{=} \int_{\mathbb{R}^r} f(\mathbf{z} - \mathbf{k}) P(\mathbf{k}) d\mathbf{k},\ P(\mathbf{k}) \stackrel{\text{def}}{=} \sum_{l=1}^{L} \mathbf{u}_l \delta(\mathbf{k} - \mathbf{r}_l) \tag{8}$$

and δ is a Dirac delta function. Thus, w as a function of $\mathbf{z}$ can be written as a convolution: $w = f \star P$. From Fourier Inversion Theorem and Convolution Theorem (see: Sec. 3.1), we thus get the following:

$$w(\mathbf{z}) = \int_{\mathbb{R}^r} \mathcal{F}_w(\xi) \exp(2\pi i \xi^\top \mathbf{z}) d\xi = \int_{\mathbb{R}^r} \mathcal{F}_f(\xi) \mathcal{F}_P(\xi) \exp(2\pi i \xi^\top \mathbf{z}) d\xi \tag{9}$$

Using any of the quadrature methods, we then approximate w as:

$$w(\mathbf{z}) \approx \tilde{w}(\mathbf{z}) = \sum_{s=1}^{S} a_s \mathcal{F}_f(\xi_s) \mathcal{F}_P(\xi_s) \exp(2\pi i \xi_s^\top \mathbf{z}) \tag{10}$$

for the quadrature coefficients $\{a_s\}_{s=1}^S$ and samples $\{\xi_s\}_{s=1}^S$. We will sample $S = O(L)$ points (in practical applications we will even take S to be a constant). Note that from the linearity of the FT and our observations on the FT of the Dirac delta function from Sec. 3.1, we get:

$$\mathcal{F}_P(\xi_s) = \sum_{l=1}^{L} \mathbf{u}_l \exp(-2\pi i \xi_s^\top \mathbf{r}_l);\ s = 1, ..., S. \tag{11}$$

Thus we can compute all $\mathcal{F}_P(\xi_s)$ for $s = 1, ..., S$ in $O(L \log(L))$ time, applying NU-FFT. With all those values efficiently calculated, we can define: $b_s = a_s \mathcal{F}_f(\xi_s) \mathcal{F}_P(\xi_s)$. Then Equation 10 becomes:

$$w(\mathbf{z}) \approx \tilde{w}(\mathbf{z}) = \sum_{s=1}^{S} b_s \exp(2\pi i \xi_s^\top \mathbf{z}) \tag{12}$$

We need to compute its value in L points: $\mathbf{z} = \mathbf{r}_1, ..., \mathbf{r}_L$. We do it again with NU-FFT, again in $O(L \log(L))$ time. That completes the algorithm. Let us call this algorithm FastMult$_\mathbf{M}$.

Note: In practical applications, quadrature coefficients $\{a_s\}_{s=1}^S$ and samples $\{\xi_s\}_{s=1}^S$ can also be made learnable. That effectively corresponds to implicitly learning the modulation function f.

Now we are ready to integrate this RPE mask $\mathbf{M}$ into Performer-style attention. Recall the low rank attention is given as follows, where $\phi : \mathbb{R}^d \to \mathbb{R}^m$ is a nonlinear (potentially randomized) map:

$$\begin{aligned} \widehat{\mathrm{Att}_{\mathrm{K}}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \widehat{\mathbf{D}}^{-1}(\phi(\mathbf{Q})(\phi(\mathbf{K})^\top \mathbf{V})), \\ \widehat{\mathbf{D}} &= \mathrm{diag}(\phi(\mathbf{Q})(\phi(\mathbf{K})^\top \mathbf{1}_L)). \end{aligned} \tag{13}$$

Here $\mathbf{1}_L$ is the all-ones vector of length L, $\mathrm{diag}(\cdot)$ is a diagonal matrix with the input vector as the diagonal and $\mathbf{Q} \in \mathbb{R}^{L \times d_{QK}}$, $\mathbf{K} \in \mathbb{R}^{L \times d_{QK}}$ and $\mathbf{V} \in \mathbb{R}^{L \times d}$ are the matrices with rows given by the sequences of queries, keys and value vectors respectively.

[11] investigated how to incorporate the masking in the linear attention as above. Note that in this case $\tilde{\mathbf{A}}$ is never materialized. Building on the work of [49], the authors of [11] proposed a general algorithm that allows for the efficient implementation of masked linear attention. For the convenience of the reader, we present this algorithm in Algorithm 2 box below. We design a mask $\mathbf{M}$ that supports fast matrix vector multiplication, which is exactly the presented above FastMult$_\mathbf{M}$ algorithm (Algorithm 1).

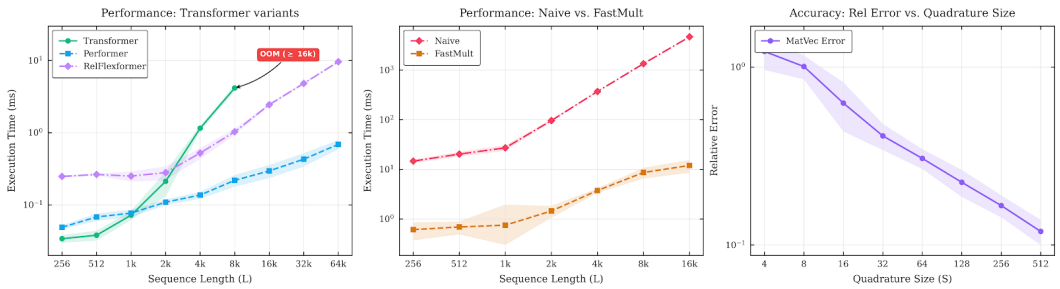


Figure 1: **Execution time (ms) as a function of sequence length (L) and accuracy of our FastMult algorithm.** *Left:* Performance comparison of Transformer, Performer, and RelFlexformer models. Note that the standard Transformer encounters an Out-of-Memory (OOM) error for sequence lengths $L \geq 16k$. *Middle:* Execution time comparison of the Naive approach versus FastMult. *Right:* Relative error of stochastic quadrature approximation to exact mask matrix-vector product as a function of the quadrature size S. Error decays as $O(1/\sqrt{S})$.

Algorithm 2: General Efficient Low-Rank Masked Attention

Input: Query/key matrices: $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{L \times d_{QK}}$, value matrix $\mathbf{V} \in \mathbb{R}^{L \times d}$, mask $\mathbf{M} \in \mathbb{R}^{L \times L}$, procedure $\text{FastMult}_{\mathbf{M}} : \mathbb{R}^L \to \mathbb{R}^L$ calculating $\mathbf{Mx}$ (or its approximation) for the input $\mathbf{x} \in \mathbb{R}^L$, kernel feature map: $\phi : \mathbb{R}^{d_{QK}} \to \mathbb{R}^m$. $\text{vec}(\cdot)$ denotes vectorization.

Output: Masked low-rank attention embeddings using ϕ.

1. Compute matrices $\mathbf{V}^1 \in \mathbb{R}^{L \times (md)}$, $\mathbf{V}^2 \in \mathbb{R}^{L \times m}$ with rows defined as:
   $\mathbf{V}^1_{i:} = \text{vec}(\phi(\mathbf{k}_i^\top)\mathbf{v}_i)$, $\mathbf{V}^2_{i:} = \phi(\mathbf{k}_i^\top)^\top$, where $\mathbf{k}_i$/$\mathbf{v}_i$ stands for the ith row of $\mathbf{K}$/$\mathbf{V}$.
2. $\tilde{\mathbf{D}}^1 := [\text{FastMult}_{\mathbf{M}}(\mathbf{V}^1_{:1}), ..., \text{FastMult}_{\mathbf{M}}(\mathbf{V}^1_{:md})] \in \mathbb{R}^{L \times md}$,
   $\tilde{\mathbf{D}}^2 := [\text{FastMult}_{\mathbf{M}}(\mathbf{V}^2_{:1}), ..., \text{FastMult}_{\mathbf{M}}(\mathbf{V}^2_{:m})] \in \mathbb{R}^{L \times m}$ for $\mathbf{V}^{1/2}_{:i}$ denoting ith column of $\mathbf{V}^{1/2}$.
3. Output the embedding $\mathbf{p}_i$ of the ith tokens as:
   $\mathbf{p}_i = \frac{\phi(\mathbf{q}_i^\top)^\top \text{devec}(\tilde{\mathbf{D}}^1_{i:})}{\phi(\mathbf{q}_i^\top)^\top (\tilde{\mathbf{D}}^2_{i:})^\top}$, where $\mathbf{q}_i$ is the ith row of $\mathbf{Q}$ and $\text{devec}(\cdot)$ devectorizes its input back to $\mathbb{R}^{m \times d}$.

Lemma 3.3 (Generalization to RoPE)**.** *For 1D sequential token coordinates, if the continuous spatial modulation function $f(z)$ is parameterized as a set of discrete harmonic functions, the NU-FFT spatial convolution perfectly recovers the discrete rotational shifts of standard Rotary Position Embedding (RoPE).*

Proof. Recall that the application of a relative positional mask is computed as a spatial convolution in the frequency domain. For a target point z, the masked value $w(z)$ is:

$$w(z) = \int_{\mathbb{R}} \mathcal{F}_f(\xi)\mathcal{F}_P(\xi) \exp(2\pi i \xi z) d\xi,$$

where $\mathcal{F}_P(\xi) = \sum_{l=1}^{L} u_l \exp(-2\pi i \xi r_l)$ is the Fourier Transform of the point cloud signal.

In standard RoPE, the dot product between queries and keys for a specific feature dimension d is modulated by a rotation matrix with base frequency θ_d. To replicate this, we define the modulation function for a single feature dimension as $f(z) = \cos(\theta_d z)$.

Taking the continuous Fourier Transform of this modulation function yields Dirac deltas at the fundamental frequencies:

$$\mathcal{F}_f(\xi) = \frac{1}{2}\left(\delta\left(\xi - \frac{\theta_d}{2\pi}\right) + \delta\left(\xi + \frac{\theta_d}{2\pi}\right)\right)$$

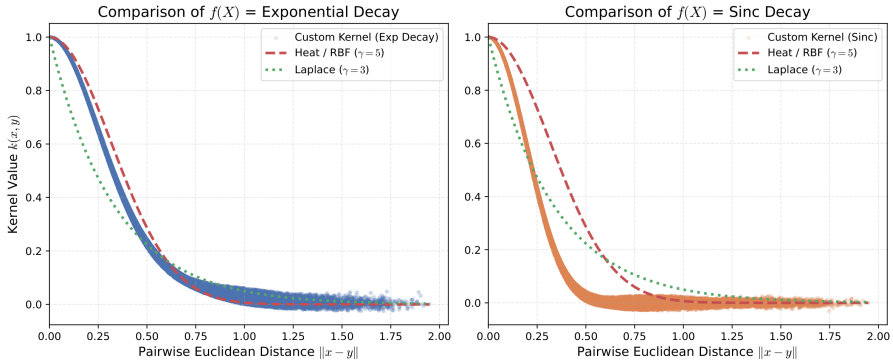


Figure 2: *RelFlexformer* **Mask Behavior Analysis.** Comparison of our proposed kernels against standard RBF and Laplace kernels. As the Euclidean distance between points increases, the kernel value decays smoothly, mimicking the behavior of Heat and Laplace kernels (dashed lines). The tight variance in the scatter highlights the stability of our projection method across varying spatial scales.

Substituting this into the convolution integral, by the properties of the Dirac delta, the integral evaluates to zero everywhere except exactly at $\pm\frac{\theta_d}{2\pi}$. The continuous integral thus collapses analytically into a discrete sum:

$$w(z) = \frac{1}{2}\mathcal{F}_P\left(\frac{\theta_d}{2\pi}\right)\exp(i\theta_d z) + \frac{1}{2}\mathcal{F}_P\left(-\frac{\theta_d}{2\pi}\right)\exp(-i\theta_d z)$$

Consequently, by fixing the quadrature sampling frequencies in Algorithm 1 to $\xi_s \in \left\{\pm\frac{\theta_d}{2\pi}\right\}$, the generalized RelFlexformer algorithm bypasses approximation and analytically evaluates the exact RoPE embedding, all without explicit materialization of the attention matrix. □

Remark (Generalization to STRING [63]). The $\otimes$-STRING position encoding computes a multiplicative mask $\frac{1}{m}\sum_{k=1}^{m}\cos(\boldsymbol{\omega}_k^\top(\mathbf{r}_i - \mathbf{r}_j))$ with *learnable* frequency vectors $\{\boldsymbol{\omega}_k\}_{k=1}^m$. Our Algorithm 1 computes the same cosine-of-projection form, by drawing $\{\boldsymbol{\omega}_k\}$ from the spectral density $\exp(-\lambda\|\mathbf{x}\|)$ (multivariate Cauchy) and keeps them *fixed*. Setting $\mathcal{F}_f(\xi_s) = 1$ and replacing the quadrature weights $\{a_s\}$ with uniform $1/S$ recovers $\otimes$-STRING with random frequencies. Conversely, making our frequencies learnable recovers $\otimes$-STRING. Thus STRING is a special case of our framework.

Figure 1 demonstrates the practical benefits of avoiding explicit materialization of the RPE mask. RelFlexformer remains tractable at sequence lengths where the standard Transformer runs out of memory, while FastMult (via Algorithm 1) substantially reduces the cost of the mask-vector product compared with the naive approach. The stochastic quadrature approximation also exhibits the expected $O(1/\sqrt{S})$ error decay, allowing the computation-accuracy trade-off to be controlled through the quadrature size S. Finally, Fig. 2 illustrates the behavior of our mask matrix. The mask exhibits a decay pattern similar to standard RBF and Laplace kernels, and its shape can be controlled via different L_1 functions and spatial scales.

4 Experiments

In this section, we describe the experimental setup and evaluate the performance of RelFlexformer. For all experiments, we set $\mathcal{F}_f(\xi) := \exp(-\lambda\|\xi\|)$, where λ is a modulation parameter. We incorporate Performer into PCT [23] for object classification (Sec. 4.1), PTv3 [84] for semantic segmentation (Sec. 4.2) and Dformer for RGB-D data (Sec. 4.3). We use standard dense self-attention as the baseline and compare it with Performer-based variants that replace dense self-attention with ReLU-Performer attention. The evaluated variants include vanilla Performer [12], Performer with PointRoPE [101], RelFlexformer, and RelFlexformer with PointRoPE. PointRoPE [101] extends RoPE to 3D point clouds in a parameter-free manner, allowing us to assess its effect both with Performer and in combination with RelFlexformer. Following [25, 59] insight that attention benefits from position

Table 1: Accuracy (%) comparison on ModelNet40 and Overall and average accuracy for ScanObjectNN classification benchmarks. Bold indicates the best score for each dataset.

Attention	ModelNet40	ScanObjectNN	
	OA	mAcc	OA
Transformer	93.2	80.5	84.0
Performer [12]	92.34	80.47	83.16
+ PointRoPE [101]	92.48	80.36	83.88
RelFlexformer (ours)	**92.94**	**81.56**	**84.45**
+ PointRoPE	92.55	81.49	84.26

Table 2: Semantic segmentation performance comparison on indoor (ScanNet, ScanNet200, and ScanNet++) and outdoor (nuScenes) benchmarks. Bold indicates the best score for each metric. RelFlexFormer replaces softmax attention (gray) with linear attention plus Fourier RPE, recovering or exceeding dense-attention accuracy at sub-quadratic cost.

Attention	ScanNet Val			ScanNet200 Val			ScanNet++ Val			nuScenes Val		
	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
Transformer	77.6	85.0	92.0	35.3	46.0	83.4	48.2	61.6	87.0	80.4	87.2	94.7
Performer [12]	74.8	83.8	91.0	28.2	39.6	80.2	48.1	**62.5**	**87.2**	72.0	80.2	93.5
+ PointRoPE [101]	74.6	83.4	91.0	34.1	44.9	**82.9**	48.1	60.9	86.9	80.4	87.4	**94.8**
RelFlexformer (ours)	**76.8**	**85.0**	**91.9**	34.0	**45.4**	**82.9**	48.7	62.2	86.8	80.3	**87.5**	94.6
+ PointRoPE [101]	76.6	84.5	91.6	**34.9**	45.2	**82.9**	**48.8**	61.7	86.7	**81.2**	**87.5**	**94.8**

information entering through multiple complementary pathways we combine RoPE (which encodes geometry into token representations) with our RPE (which modulates interaction strength by distance).

For RGBD data, we also compare against the recently introduced STRING [63]. We provide a description for all the datasets used for evaluation in App. B.1. Finally we provide extensive ablation studies showing only a small quadrature size is optimal for various tasks C.

4.1 Object Classification

We evaluate object classification performance of PCT on ModelNet40 [87] and ScanObjectNN [74]. On both ModelNet40 and ScanObjectNN, Performer underperforms the Transformer baseline, indicating a performance gap introduced by efficient attention. Augmenting Performer with PointRoPE yields only marginal gains on both the datasets. RelFlexformer improves performance to **92.9**%, reducing most of this gap on ModelNet40. On ScanObjectNN, our method *surpasses the dense Transformer baseline* (see Tables 14 and 15 for more comparisons with prior methods).

4.2 RelFlexformer for Semantic Segmentation

We implement RelFlexformer on top of PTv3 [84] and follow the training protocol of Pointcept [54]. Tables 2 and 3 compare different attention mechanisms within the PTv3 framework. All variants are evaluated under the full fine-tuning setting on semantic segmentation benchmarks, including ScanNet [15], ScanNet200 [62], ScanNet++ [96], nuScenes [5], and S3DIS [1]. For additional comparisons with prior methods, please see Tables 16–18.

Table 2 shows that replacing Transformer with Performer reduces mIoU across all datasets. Because Performer [12] approximates the softmax kernel using random features, it may be less effective at capturing fine-grained spatial interactions in irregular 3D point clouds. While PointRoPE improves Performer on ScanNet200, nuScenes, and most reported S3DIS areas, the improvements are not consistent across benchmarks. In particular, it provides no improvement or slightly degrades mIoU on ScanNet, ScanNet++, and S3DIS Area 2. These results suggest that RoPE-style 3D positional encoding can benefit Performer, while the non-uniform gains highlight the importance of additional relative positional modulation for capturing fine-grained geometric interactions.

RelFlexformer largely resolves this performance drop by introducing NU-FFT-driven RPE modulation into efficient attention. Compared with vanilla Performer, RelFlexformer improves mIoU from 74.8

Table 3: Semantic segmentation performance comparison on S3DIS under 6-fold cross-validation. Bold indicates the best score for each metric. RelFlexFormer matches or exceeds dense-attention performance (gray) on 6-fold cross validation, closing the gap left by Performer alone.

Attention	Metric	Area1	Area2	Area3	Area4	Area5	Area6	6-Fold
Transformer	allAcc	93.22	86.26	94.56	90.72	91.67	94.98	91.90
	mAcc	89.92	74.44	94.45	81.11	78.92	93.55	85.31
	mIoU	83.01	63.42	86.66	71.34	73.43	87.31	77.70
Performer [12]	allAcc	92.35	**88.53**	94.47	85.96	90.86	91.20	90.56
	mAcc	88.56	76.97	**93.69**	78.31	75.63	85.77	83.16
	mIoU	80.74	62.60	86.02	62.60	69.75	77.35	73.18
+ PointRoPE [101]	allAcc	93.00	86.88	94.25	88.29	91.17	94.73	91.39
	mAcc	90.06	76.62	93.63	76.14	**77.55**	93.26	84.54
	mIoU	82.31	61.47	86.21	67.38	71.75	86.59	75.95
RelFlexformer (ours)	allAcc	92.96	88.07	94.51	88.36	91.14	94.65	**91.62**
	mAcc	**90.59**	**77.33**	93.00	80.92	76.79	**93.33**	**85.33**
	mIoU	81.93	**63.21**	86.20	**68.28**	71.01	**86.88**	76.25
+ PointRoPE [101]	allAcc	**93.32**	85.84	**94.57**	**88.89**	**91.50**	**94.74**	91.48
	mAcc	90.43	73.82	93.64	**81.59**	77.44	93.14	85.01
	mIoU	**82.79**	62.87	**86.75**	68.12	**72.14**	86.79	**76.58**

to **76.8** on ScanNet, from 28.2 to **34.0** on ScanNet200, from 48.1 to **48.7** on ScanNet++, and from 72.0 to **80.3** on nuScenes. Table 3 further shows that RelFlexformer improves mIoU over Performer on all six S3DIS held-out areas, with gains ranging from 0.18 to 9.53 points. The 6-fold mIoU also increases from 73.18 to **76.25**. The observed improvements demonstrate that the proposed RPE modulation substantially strengthens Performer-style efficient attention. It consistently improves over vanilla Performer, closes much of the mIoU gap to dense Transformer attention, and *even surpasses the Transformer on ScanNet++*.

We further examine the interaction between PointRoPE and RelFlexformer. Across all datasets, RelFlexformer with PointRoPE achieves higher mIoU than Performer with PointRoPE, showing that the proposed RPE modulation provides additional benefits beyond token-level 3D positional encoding. For example, mIoU increases from 34.1 to **34.9** on ScanNet200, from 48.1 to **48.8** on ScanNet++, and from 80.4 to **81.2** on nuScenes. Similar improvements are also observed on ScanNet and S3DIS. As a result, the combined model approaches the dense Transformer on ScanNet, ScanNet200, and most reported S3DIS areas, and *even surpasses it on ScanNet++, nuScenes, and S3DIS Area 3*.

4.3 RGB-D experiments

In this section, we demonstrate that RelFlexFormer integrates effectively with state-of-the-art architectures such as DFormer [99] for RGB-D semantic segmentation. Starting from a pre-trained DFormer-Base, we replace dense attention with ReLU-Performers and fine-tune on NYU Depth v2 [67] and SUN RGB-D (Table 4). RelFlexFormer consistently outperforms the Performer baseline and substantially narrows the gap to dense attention. On the challenging SUN RGB-D dataset with multisensor inputs, Performer exhibits a notable performance gap relative to the Transformer. Our method substantially improves performance and effectively closes this gap. Furthermore, we compare against the recently proposed RPE method STRING [63]. Our approach outperforms STRING while avoiding its additional parameter overhead ($\sim$ 70K parameters). For additional comparisons with prior methods, please see Tables 19 and 20.

Additional efficiency benchmarks are provided in Appendix E. Across PTv3 and DFormer, RelFlexformer substantially reduces memory and computational cost relative to Softmax attention while retaining relative positional encoding, and exhibits favorable scaling as the number of tokens increases.

Table 4: **RGB-D semantic segmentation results** (mIoU %) on NYU Depth v2 and SUN RGB-D. All methods use DFormer-Base with HAM decoder. *Performer* replaces softmax with ReLU linear attention. RelFlexformer is the best sub-quadratic variant and substantially closes the gap with dense Transformer.

Attention	Complexity	NYU v2	SUN RGB-D
Softmax	$O(n^2)$	55.6	51.2
Performer	$O(n)$	54.44	48.49
Performer + STRING [63]	$O(n \log n)$	54.96	50.87
RelFlexFormer (ours)	$O(n \log n)$	**55.32**	**51.04**

4.4 RPE Improves Dense Attention

To verify whether performance gains stem from the kernel approximation or positional encoding, we add RelFlexformer RPE directly into standard dense Softmax Transformers. As shown in Table 5, introducing our RPE consistently improves dense attention across all datasets, confirming the gains are driven by the **geometric prior**.

Table 5: **Impact of RelFlexformer RPE on dense Transformers.** Evaluated across various 3D modeling datasets, RelFlexFormer improves the dense Transformers.

Dataset	Transformer	Transformer + RelFlexformer RPE
ScanObjectNN (OA)	84.0	**84.98**
ModelNet40 (OA)	92.2	**92.46**
NYU (mIoU, MST)	55.6	**56.32**
SUN RGB-D (mIoU, MST)	51.2	**51.40**
ScanNet200 (mIoU)	35.3	**36.50**

In summary, we show that RelFlexFormer significantly improves upon the Performer baseline, closing the gap with dense attention across all datasets and even surpassing Transformers on ScanObjectNN, S3DIS, nuScenes, and ScanNet++. The geometric inductive bias from RelFlexFormer also improves the regular Transformers as well. Furthermore, when PointRoPE is competitive, combining it with RelFlexFormer yields additional gains, demonstrating that our architecture can effectively integrate multiple complementary inductive biases.

5 Conclusion

RelFlexformers provide a new class of efficient 3D Transformers that combine expressive relative positional encodings with subquadratic attention. By framing geometric modulation via the Non-Uniform Fourier Transform, we have shown that it is possible to incorporate a broad family of integrable RPE functions without sacrificing the computational benefits of kernelized attention. Our approach achieves a complexity of $O(L \log L)$, establishing it as a highly scalable architecture for processing large-scale, irregular point clouds that were previously bottlenecked by the quadratic cost of standard dense attention. Our empirical results across various 3D benchmarks demonstrate that the flexibility provided by our framework, specifically the ability to handle non-structured, heterogeneous token distributions, leads to consistent performance gains over the baseline Performer and even surpassing the dense Transformers.

Acknowledgements

BK and MO were supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. RS-2022-NR071853, RS-2023-00222663, RS-2026-25613008).

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1534–1543, 2016.

[2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[3] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. Multimae: Multi-modal multi-task masked autoencoders. In *European conference on computer vision*, pages 348–367. Springer, 2022.

[4] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[6] Jinming Cao, Hanchao Leng, Dani Lischinski, Daniel Cohen-Or, Changhe Tu, and Yangyan Li. Shapeconv: Shape-aware convolutional layer for indoor rgb-d semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7088–7097, 2021.

[7] Lin-Zhuo Chen, Zheng Lin, Ziqin Wang, Yong-Liang Yang, and Ming-Ming Cheng. Spatial information guided convolution for real-time rgbd semantic segmentation. *IEEE Transactions on Image Processing*, 30:2313–2324, 2021.

[8] Xiaokang Chen, Kwan-Yee Lin, Jingbo Wang, Wayne Wu, Chen Qian, Hongsheng Li, and Gang Zeng. Bi-directional cross-modality feature propagation with separation-and-aggregation gate for rgb-d semantic segmentation. In *European conference on computer vision*, pages 561–577. Springer, 2020.

[9] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Largekernel3d: Scaling up kernels in 3d sparse cnns. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13488–13498. IEEE, 2023.

[10] Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. In *2019 International Conference on 3D Vision (3DV)*, pages 155–163. IEEE, 2019.

[11] Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamás Sarlós, Adrian Weller, and Thomas Weingarten. From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research, pages 3962–3983. PMLR, 2022.

[12] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[13] Krzysztof Marcin Choromanski, Arijit Sehanobish, Somnath Basu Roy Chowdhury, Han Lin, Kumar Avinava Dubey, Tamas Sarlos, and Snigdha Chaturvedi. Fast tree-field integrators: From low displacement rank to topological transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019.

[15] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[17] Nathan Doumèche, Francis Bach, Gérard Biau, and Claire Boyer. Fast kernel methods: Sobolev, physics-informed, and additive models. In *Forty-third International Conference on Machine Learning*, 2026.

[18] Siqi Du, Weixi Wang, Renzhong Guo, Ruisheng Wang, and Shengjun Tang. Asymformer: Asymmetrical cross-modal representation learning for mobile platform real-time rgb-d semantic segmentation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7608–7615. IEEE, 2024.

[19] Gerald B Folland. *Fourier analysis and its applications*, volume 4. American Mathematical Soc., 2009.

[20] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens Van Der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A single model for many visual modalities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16102–16112, 2022.

[21] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[22] Leslie Greengard and June-Yub Lee. Accelerating the nonuniform fast fourier transform. *SIAM Review*, 46(3):443–454, 2004.

[23] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, Apr 2021.

[24] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European conference on computer vision*, pages 345–360. Springer, 2014.

[25] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

[26] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[27] Xinxin Hu, Kailun Yang, Lei Fei, and Kaiwei Wang. Acnet: Attention based network to exploit complementary features for rgbd semantic segmentation. In *2019 IEEE international conference on image processing (ICIP)*, pages 1440–1444. IEEE, 2019.

[28] Ermo Hua, Che Jiang, Xingtai Lv, Kaiyan Zhang, Youbang Sun, Yuchen Fan, Xuekai Zhu, Biqing Qi, Ning Ding, and Bowen Zhou. Fourier position embedding: enhancing attention's periodic extension for length generalization. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

[29] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[30] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[31] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[32] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[33] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

[34] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8500–8509, 2022.

[35] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[36] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018.

[37] Huan Lei, Naveed Akhtar, and Ajmal Mian. Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[38] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

[39] Zhuoyuan Li, Yubo Ai, Jiahao Lu, ChuXin Wang, Jiacheng Deng, Hanzhi Chang, Yanzhe Liang, Wenfei Yang, Shifeng Zhang, and Tianzhu Zhang. Pamba: enhancing global interaction in point clouds via state space model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5092–5100, 2025.

[40] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. *Advances in neural information processing systems*, 37:32653–32677, 2024.

[41] Haojia Lin, Xiawu Zheng, Lijiang Li, Fei Chao, Shanshan Wang, Yan Wang, Yonghong Tian, and Rongrong Ji. Meta architecture for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17682–17691, 2023.

[42] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *European Conference on Computer Vision*, pages 657–675. Springer, 2022.

[43] Jiuming Liu, Ruiji Yu, Yian Wang, Yu Zheng, Tianchen Deng, Weicai Ye, and Hesheng Wang. Point mamba: A novel point cloud backbone based on state space model with octree-based ordering strategy. *arXiv preprint arXiv:2403.06467*, 2024.

[44] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI conference on artificial intelligence*, pages 8778–8785, 2019.

[45] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8895–8904, 2019.

[46] Youquan Liu, Lingdong Kong, Xiaoyang Wu, Runnan Chen, Xin Li, Liang Pan, Ziwei Liu, and Yuexin Ma. Multi-space alignments towards universal lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

[47] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[48] Dening Lu, Qian Xie, Mingqiang Wei, Kyle Gao, Linlin Xu, and Jonathan Li. Transformers in 3d point clouds: A survey. *arXiv preprint arXiv:2205.07417*, 2022.

[49] Shengjie Luo, Shanda Li, Tianle Cai, Di He, Dinglan Peng, Shuxin Zheng, Guolin Ke, Liwei Wang, and Tie-Yan Liu. Stable, fast and accurate: Kernelized attention with relative positional encoding. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22795–22807. Curran Associates, Inc., 2021.

[50] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

[51] Yatian Pang, Eng Hock Francis Tay, Li Yuan, and Zhenghua Chen. Masked autoencoders for 3d point cloud self-supervised learning. *World Scientific Annual Review of Artificial Intelligence*, 1:2440001, 2023.

[52] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16928–16937. IEEE, 2022.

[53] Bohao Peng, Xiaoyang Wu, Li Jiang, Yukang Chen, Hengshuang Zhao, Zhuotao Tian, and Jiaya Jia. Oa-cnns: Omni-adaptive sparse cnns for 3d semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

[54] Pointcept Contributors. Pointcept: A codebase for point cloud perception research. `https://github.com/Pointcept/Pointcept`, 2023.

[55] Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

[56] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[57] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[58] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in neural information processing systems*, 35:23192–23204, 2022.

[59] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020.

[60] Isaac Reid, Kumar Avinava Dubey, Deepali Jain, William F Whitney, Amr Ahmed, Joshua Ainslie, Alex Bewley, Mithun George Jacob, Aranyak Mehta, David Rendleman, Connor Schenck, Richard E. Turner, René Wagner, Adrian Weller, and Krzysztof Marcin Choromanski. Linear transformer topological masking with graph random features. In *The Thirteenth International Conference on Learning Representations*, 2025.

[61] Damien Robert, Hugo Raguet, and Loic Landrieu. Efficient 3d semantic segmentation with superpoint transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[62] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *European conference on computer vision*, pages 125–141. Springer, 2022.

[63] Connor Schenck, Isaac Reid, Mithun George Jacob, Alex Bewley, Joshua Ainslie, David Rendleman, Deepali Jain, Mohit Sharma, Kumar Avinava Dubey, Ayzaan Wahid, Sumeet Singh, René Wagner, Tianli Ding, Chuyuan Fu, Arunkumar Byravan, Jake Varley, Alexey A. Gritsenko, Matthias Minderer, Dmitry Kalashnikov, Jonathan Tompson, Vikas Sindhwani, and Krzysztof Marcin Choromanski. Learning the roPEs: Better 2d and 3d position encodings with STRING. In *Forty-second International Conference on Machine Learning*, 2025.

[64] Daniel Seichter, Söhnke Benedikt Fischedick, Mona Köhler, and Horst-Michael Groß. Efficient multi-task rgb-d scene analysis for indoor environments. In *2022 International joint conference on neural networks (IJCNN)*, pages 1–10. IEEE, 2022.

[65] Daniel Seichter, Mona Köhler, Benjamin Lewandowski, Tim Wengefeld, and Horst-Michael Gross. Efficient rgb-d semantic segmentation for indoor scene analysis. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 13525–13531. IEEE, 2021.

[66] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *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.

[67] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[68] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[69] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024.

[70] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, 2020.

[71] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[72] Lyne Tchapmi, Christopher Choy, Iro Armeni, Jun Young Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2017.

[73] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[74] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[76] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[77] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *ACM Transactions on Graphics (TOG)*, 42(4):1–11, 2023.

[78] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[79] Yikai Wang, Xinghao Chen, Lele Cao, Wenbing Huang, Fuchun Sun, and Yunhe Wang. Multimodal token fusion for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12186–12195, 2022.

[80] Yikai Wang, Wenbing Huang, Fuchun Sun, Tingyang Xu, Yu Rong, and Junzhou Huang. Deep multimodal fusion by channel exchanging. *Advances in neural information processing systems*, 33:4835–4845, 2020.

[81] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. In *ACM Transactions on Graphics (SIGGRAPH Asia) 38(5)*, pages 146:1–146:12, 2019.

[82] Wenxuan Wu, Li Fuxin, and Qi Shan. Pointconvformer: Revenge of the point-based convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21802–21813, 2023.

[83] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019.

[84] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4840–4851, 2024.

[85] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

[86] Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

[87] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[88] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4606–4615, 2018.

[89] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

[90] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European conference on computer vision (ECCV)*, pages 87–102, 2018.

[91] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European conference on computer vision*, 2022.

[92] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5589–5598, 2020.

[93] Fan Yang, Lin Guo, Zhi Chen, and Wenbing Tao. One inlier is first: Towards efficient position encoding for point cloud registration. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[94] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[95] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *Computational Visual Media*, 11(1):83–101, 2025.

[96] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[97] Bo-Wen Yin, Jiao-Long Cao, Ming-Ming Cheng, and Qibin Hou. Dformerv2: Geometry self-attention for rgbd semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19345–19355, 2025.

[98] Bo-Wen Yin, Jiao-Long Cao, Xuying Zhang, Yuming Chen, Ming-Ming Cheng, and Qibin Hou. Omnisegmentor: A flexible multi-modal learning framework for semantic segmentation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[99] Bowen Yin, Xuying Zhang, Zhong-Yu Li, Li Liu, Ming-Ming Cheng, and Qibin Hou. DFormer: Rethinking RGBD representation learning for semantic segmentation. In *The Twelfth International Conference on Learning Representations*, 2024.

[100] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022.

[101] Yuanwen Yue, Damien Robert, Jianyuan Wang, Sunghwan Hong, Jan Dirk Wegner, Christian Rupprecht, and Konrad Schindler. Litept: Lighter yet stronger point transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24173–24184, 2026.

[102] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: transformers for longer sequences. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS 20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[103] Jiaming Zhang, Huayao Liu, Kailun Yang, Xinxin Hu, Ruiping Liu, and Rainer Stiefelhagen. Cmx: Cross-modal fusion for rgb-x semantic segmentation with transformers. *IEEE Transactions on intelligent transportation systems*, 24(12):14679–14694, 2023.

[104] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022.

[105] Tao Zhang, Haobo Yuan, Lu Qi, Jiangning Zhang, Qianyu Zhou, Shunping Ji, Shuicheng Yan, and Xiangtai Li. Point cloud mamba: Point cloud learning via state space model. In *Proceedings of the AAAI conference on artificial intelligence*, pages 10121–10130, 2025.

[106] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[107] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

[108] Wujie Zhou, Enquan Yang, Jingsheng Lei, Jian Wan, and Lu Yu. Pgdenet: Progressive guided fusion and depth enhancement network for rgb-d indoor scene parsing. *IEEE Transactions on Multimedia*, 25:3483–3494, 2022.

[109] Wujie Zhou, Enquan Yang, Jingsheng Lei, and Lu Yu. Frnet: Feature reconstruction network for rgb-d indoor scene parsing. *IEEE Journal of Selected Topics in Signal Processing*, 16(4):677–687, 2022.

[110] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

A Additional Related Work

In this section we discuss additional related works mostly focusing on depth images and then discuss differences between our method against other RPE methods.

A.1 Related Work on Depth Images

The integration of depth information into semantic segmentation has evolved from simple heuristic-based approaches to sophisticated geometric modeling. Early methods, such as those by [24], utilized HHA encoding to map depth into a geocentric coordinate system, enabling the use of 2D CNNs pretrained on RGB datasets. However, these methods often suffered from a "modality gap" where the network failed to learn cross-modal dependencies effectively.

The introduction of the Vision Transformer (ViT) [16] facilitated more flexible fusion strategies. Early ViT-based architectures like TokenFusion [79] and CMX [103] focused on cross-modal feature alignment through shared attention mechanisms. Furthermore, hierarchical architectures such as the Swin Transformer [47] introduced shifted-window attention to capture multi-scale dependencies, which has become a staple for dense prediction tasks. Despite their success, these models predominantly treat the depth map as a supplementary 2D image, often neglecting the inherent 3D geometric priors of the scene.

A significant paradigm shift occurred with the move toward coordinate lifting, where 2D pixels are unprojected into a 3D camera coordinate space (x, y, z). DFormer [99] and its successor DFormerv2 [97] represent milestones in this direction, rethinking RGB-D representation by treating depth maps as geometry priors rather than simple texture maps. By leveraging camera intrinsics (f_x, f_y, c_x, c_y), lifting-based methods enable the model to reason about physical scale and spatial proximity. Our work extends this geometric intuition; while recent frameworks like OmniSegmentor [98] explore flexible multi-modal learning, we argue that capturing the interaction between lifted tokens requires a nuanced understanding of relative spatial distances in 3D space. Unlike 2D-centric models, we operate on a unified geometric representation. This allows our proposed RPE to transcend the 2D pixel grid, effectively unifying the geometric reasoning required for both sparse point cloud representations and dense RGB-D lifting within a single, geometry-agnostic framework.

Relative Positional Encoding (RPE) allows the attention mechanism to attend to the displacement between tokens, a concept popularized in 2D by Swin Transformer [47]. However, 2D RPE is typically constrained to discrete indices. Recent work on 3D-aware transformers [107, 84] has sought to inject 3D structural integrity into the attention kernel. Most notably, STRING [63] introduced a family of separable, translation-invariant position encodings that generalize Rotary Position Encodings (RoPE) to multidimensional coordinates using a Lie group framework. In this work, we propose a generalized 3D RPE where STRING can be viewed as a specific instantiation. By defining our RPE as a more general operator over Euclidean space, we ensure that the attention mechanism is biased by actual physical distance, which is critical for resolving depth-dependent occlusions and complex 3D layouts across diverse data modalities.

While 3D-aware geometric reasoning provides superior accuracy, it is inherently limited by the $O(N^2)$ computational complexity of standard attention. This bottleneck is acute in high-resolution RGB-D segmentation. To maintain tractability, we adopt the Performer architecture [12], utilizing a linear-time approximation of the attention matrix. This allows our model to scale to the high token counts required for dense 3D lifting, achieving a balance of geometric fidelity and efficiency that is often unattainable for standard quadratic Transformers.

A.2 Comparison with Fast Masked Attention Methods

Our masked linear-attention formulation builds on the general observation that a masked attention operator can be evaluated efficiently when the mask admits fast matrix-vector multiplication [11]. The distinction is in how the mask is represented and evaluated. Prior approaches obtain fast multiplication by exploiting structure in the token domain, whereas RelFlexformer constructs the geometric operator directly from arbitrary non-equispaced coordinates.

Block-Toeplitz masks. The Block-Toeplitz formulation [11] enables fast mask multiplication through FFTs, but its efficiency relies on Toeplitz structure induced by equispaced token locations.

This structure is not available for general point clouds, where coordinates are non-equispaced. Applying the same construction therefore requires first mapping the points to a regular grid, which changes the underlying geometry. In contrast, RelFlexformer uses NU-FFT to evaluate the corresponding Fourier representation directly at the input coordinates. When the coordinates lie on a regular grid, our formulation reduces to the conventional Toeplitz/FFT computation as a special case.

Tree-based masks. FTFI [13] extends fast masked attention to irregular data by constructing a tree-based representation of the mask. This provides an alternative for non-grid data, but requires explicit construction of the auxiliary tree. This distinction is particularly important for point-cloud Transformers whose token sets can change during training. For example, PCT operates on a subsampled set of 256 points selected from 1024 points using coordinate and feature similarity. Since the selected tokens depend on learned features, the token set changes during optimization, requiring the tree structure to be reconstructed rather than amortized over training. RelFlexformer requires neither a graph nor a tree and operates directly on the current coordinates through NU-FFT.

We measure this difference on the PCT token sets, where FTFI's tree construction requires approximately 14,870 ms per optimization step, compared with 17.4 ms for the RelFlexformer attention computation. Thus, although both methods support irregular token locations, their computational costs differ substantially when the token set changes dynamically. This recurring preprocessing cost is substantial relative to the overall training time of DFormer, which takes approximately 6 hours on NYUv2 and over 24 hours on SUN RGB-D.

B Implementation Details

In this section we will describe the datasets and our implementations for the point cloud based tasks and the RGB-D datasets. To provide a strong and widely used efficient-attention baseline, we additionally implement Performer [12] across our point cloud and RGB-D experiments. Standard dot-product self-attention explicitly computes pairwise interactions between all input tokens, resulting in quadratic complexity with respect to the number of tokens. Given query, key, and value matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L\times d}$, dense self-attention is computed as

$$\mathrm{Att}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V}, \quad \mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^\top/\sqrt{d}), \quad \mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_L). \tag{14}$$

Although this formulation is expressive, the cost of constructing the $L \times L$ attention matrix becomes expensive for dense point clouds and high-resolution RGB-D inputs.

Performer approximates softmax attention using a positive random feature map $\phi : \mathbb{R}^d \to \mathbb{R}^m$.

$$\widehat{\mathrm{Att}}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \widehat{\mathbf{D}}^{-1}(\phi(\mathbf{Q})(\phi(\mathbf{K})^\top\mathbf{V})), \quad \widehat{\mathbf{D}} = \mathrm{diag}(\phi(\mathbf{Q})(\phi(\mathbf{K})^\top\mathbf{1}_L)). \tag{15}$$

This avoids explicitly materializing the full attention matrix and reduces the attention complexity from quadratic to linear in L, up to the number of random features. Therefore, Performer serves as a suitable baseline for evaluating whether RelFlexformer improves over existing linear-attention mechanisms in 3D perception settings. In our work, ϕ =ReLU.

For fair comparison, we integrate Performer by replacing only the dense self-attention module in each corresponding backbone, while keeping the remaining architecture, input preprocessing, data augmentation, optimizer, and training schedule unchanged unless otherwise specified. We use fixed projection matrices so our mask adds no additional parameters.

B.1 Datasets

To evaluate the versatility and scalability of RelFlexformer, we conduct experiments across a diverse set of 3D benchmarks:

- **ModelNet40 [87]:** A synthetic object classification dataset with 12,311 CAD models from 40 categories, split into 9,843 training and 2,468 testing samples.

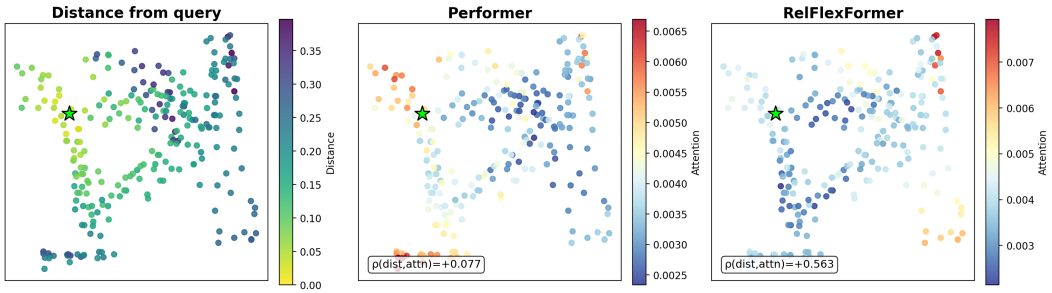


Figure 3: Distance-attention correlation. The left panel shows the spatial distance from the selected query point, while the middle and right panels visualize the corresponding attention values for Performer and RelFlexformer. Compared with Performer, RelFlexformer shows a stronger positive relationship between spatial distance suggesting that its RPE-modulated attention better reflects geometric organization around the query point.

- **ScanObjectNN [74]:** A real-world object classification dataset with about 15,000 scanned objects across 15 categories, featuring occlusion, partial scans, and background clutter.
- **S3DIS [1]:** A large-scale indoor semantic segmentation dataset with six areas and 13 classes. We follow the standard 6-fold cross-validation protocol.
- **ScanNet v2 [15]:** A large-scale indoor RGB-D dataset with 20 semantic categories. We use the standard split of 1,201 training and 312 validation scans.
- **ScanNet200 [62]:** An extension of ScanNet v2 with 200 fine-grained categories, designed to evaluate robustness under long-tailed class distributions.
- **ScanNet++ [96]:** A high-fidelity indoor 3D benchmark with accurate geometry, RGB-D captures, and fine-grained semantic annotations.
- **nuScenes [5]:** A large-scale outdoor autonomous driving dataset with 1,000 urban scenes. We use it to evaluate RelFlexformer on sparse LiDAR point clouds across 16 classes.
- **NYU Depth v2 [67]:** 1,449 indoor RGB-D scenes (795 train / 654 test) from a single Kinect v1 sensor, 40 semantic classes, 480×640 resolution.
- **SUN RGB-D [68]:** 10,335 indoor RGB-D scenes (5,285 train / 5,050 test) from 4 different sensors (Kinect v1/v2, Xtion, RealSense), 37 semantic classes. The multi-sensor variation makes position encoding harder due to varying intrinsics and depth scales.

B.2 Point Cloud Based Tasks.

In this section, we will describe our setup and hyperparameters for point cloud classification and segmentation.

B.2.1 Classification

ModelNet40. Our implementation is based on the official PCT [23] codebase. We evaluate on ModelNet40, which consists of 12,311 CAD models from 40 object categories, split into 9,843 training and 2,468 test objects. We uniformly sample 1,024 points from each object.

We use the PCT classification backbone. The model first applies neighbor embedding based on farthest-point sampling and k-NN grouping to capture local geometric structures. The embedded point features are then processed by four stacked efficient attention (Performer) layers with feature dimension d=256 and 4 attention heads. We concatenate the outputs of the four attention layers and apply a linear projection. Global max pooling and average pooling are then concatenated to obtain an object-level representation, which is passed to an MLP classification head for 40-way classification.

We optimize with SGD using momentum 0.9 and weight decay 5×10^{-4}. The initial learning rate is 0.01 and is scheduled with cosine annealing over 250 epochs. Batch size is 32.

Training augmentations include random point dropout, random rotation around the upright axis, uniform scaling, random shift, jittering, and point shuffling.

For RPE, we sample $S(=8)$ frequency vectors from a random normal distribution at initialization and keep them fixed throughout training. We set $\lambda = 1$. All experiments run on NVIDIA H100 80GB HBM3 Tensor Core GPUs.

Fig. 3 illustrates RelFlexformer shows a stronger positive relationship between spatial distance and attention patterns than Performer, suggesting that its RPE-modulated attention better reflects geometric organization around the query point.

ScanObjectNN. We evaluate on the PB_T50_RS variant (hardest split): 2,902 training and 581 test objects across 15 classes, consisting of real-world scans with background clutter and partial views. We subsample 1,024 points per object.

We use the PCT backbone: farthest-point sampling with k-NN grouping, four self-attention layers (d=256, 4 heads), global max+average pooling, and an MLP classification head ($256 \to 256 \to 15$) with dropout 0.5.

We optimize with SGD (momentum 0.9, weight decay 5×10^{-4}) using an initial learning rate of 0.01 with cosine annealing over 250 epochs. Batch size is 32.

Training augmentations include random point dropout (up to 87.5%), random y-axis rotation, uniform scaling $[0.8, 1.25]$, random shift ±0.1, jitter (σ=0.01, clip 0.02), and point shuffle.

For RPE, we sample S frequency vectors from a random normal distribution at initialization and keep them fixed throughout training. We ablate $S \in \{16, 24, 32, 64, 128, 256\}$ and $\lambda \in \{0.5, 0.75, 1, 1.5, 1.75, 2, 2.25, 3\}$. NVIDIA RTX PRO 6000 (Blackwell) GPUs with 96 GB VRAM.

B.2.2 Segmentation

For experiments based on Pointcept [54], we use its official PTv3 implementation. Since our Performer-based variants replace dense Transformer attention with Performer attention, we do not employ FlashAttention, which is designed for dense attention computation. Accordingly, in the PTv3-based experiments, we set the patch size of both the encoder and decoder to 128. Also, Z, TZ, H, and TH denote Z-order, Trans Z-order, Hilbert order, and Trans Hilbert order, respectively.

RelFlexformer is implemented by introducing a continuous relative positional mask into Performer attention. For all RelFlexformer experiments, we define the Fourier-domain modulation function as $\mathcal{F}_f(\xi) := \exp(-\lambda\|\xi\|)$, where λ is a modulation parameter. Unless otherwise specified, the quadrature size S used in the NU-FFT-based approximation of the RelFlexformer relative positional mask is fixed to 8.

For the PointRoPE variants, we adjust the channel and head configurations to make the per-head feature dimension compatible with the 3D RoPE decomposition. PointRoPE [101] splits each attention-head feature into three axis-wise subspaces corresponding to the x, y, and z coordinates and applies 1D RoPE independently to each subspace. Since RoPE applies pair-wise rotations within each axis, the per-head dimension should be divisible by 6. Following the LitePT-S* [101] configuration, we set the channel and head dimensions of the PointRoPE attention so that the per-head dimension is 18.

The detailed configurations are provided in Tables 6 and 7. All experiments run on 4×NVIDIA H100 80GB HBM3 Tensor Core GPUs. We use FP32 training throughout.

Table 6: Model configurations for PTv3-based Performer and RelFlexformer variants without PointRoPE.

Config	Value
Variants without PointRoPE	
Serialization pattern	Z + TZ + H + TH
Patch interaction	Shift Order + Shuffle Order
Positional encoding	xCPE
Embedding depth	2
Embedding channels	32
Encoder depth	[2, 2, 6, 2]
Encoder channels	[64, 128, 256, 512]
Encoder heads	[4, 8, 16, 32]
Encoder patch size	[128, 128, 128, 128]
Decoder depth	[2, 2, 2, 2]
Decoder channels	[64, 64, 128, 256]
Decoder heads	[4, 4, 8, 16]
Decoder patch size	[128, 128, 128, 128]
Down stride	[$\times$2, $\times$2, $\times$2, $\times$2]
MLP ratio	4
QKV bias	True
Drop path	0.3
Backbone output channels	64
RelFlexformer-specific	
Number of sampled frequencies (S)	8
Modulation parameter (λ)	$\{1.0, 2.0, 3.0\}$

Table 7: Model configurations for PTv3-based Performer and RelFlexformer variants with PointRoPE.

Config	Value
Variants with PointRoPE	
Serialization pattern	Z + TZ + H + TH
Patch interaction	Shift Order + Shuffle Order
Positional encoding	xCPE + PointRoPE
Embedding depth	2
Embedding channels	32
Encoder depth	[2, 2, 6, 2]
Encoder channels	[72, 144, 252, 504]
Encoder heads	[4, 8, 14, 28]
Encoder patch size	[128, 128, 128, 128]
Encoder PointRoPE frequency	[100, 100, 100, 100]
Decoder depth	[2, 2, 2, 2]
Decoder channels	[72, 72, 144, 252]
Decoder heads	[4, 4, 8, 14]
Decoder patch size	[128, 128, 128, 128]
Decoder PointRoPE frequency	[100, 100, 100, 100]
Down stride	[$\times$2, $\times$2, $\times$2, $\times$2]
MLP ratio	4
QKV bias	True
Drop path	0.3
Backbone output channels	72
RelFlexformer-specific	
Number of sampled frequencies (S)	8
Modulation parameter (λ)	$\{1.0, 2.0, 3.0\}$

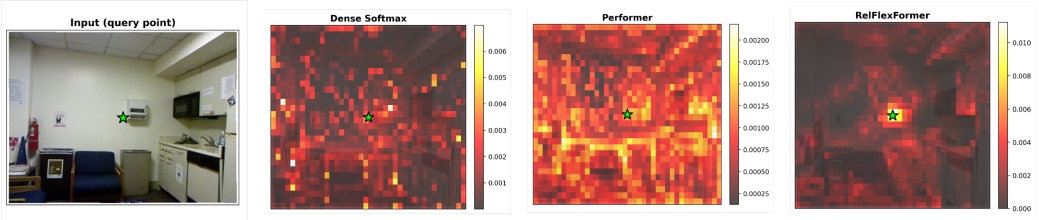


Figure 4: RelFlexformer preserves the efficiency of linear attention while producing more localized, geometry-aware attention patterns in 3D scenes.

B.3 Segmentation on RGB-D datasets

All experiments use DFormer-Base as the backbone with a HAM decoder (embedding dim 512). The *Performer* variant replaces softmax attention with ReLU-based linear attention [12] in the window attention path. *RelFlexFormer* augments Performer attention with a RPE mask. All models initialize from official DFormer-Base pretrained weights (trained with softmax attention). The full backbone is fine-tuned end-to-end.

Hyperparameters. Table 8 presents the detailed training configurations for RGB-D datasets.

Table 8: **Training configuration** for NYU Depth v2 and SUN RGB-D.

	NYU v2	**SUN RGB-D**
Optimizer	AdamW	AdamW
Learning rate	6×10^{-5}	8×10^{-5}
LR schedule	Poly (p=0.9)	Poly (p=0.9)
Weight decay	0.01	0.01
Batch size	8	16
Epochs	500	300
LR warmup epochs	10	10
Drop path rate	0.1	0.1
Fourier RPE		
S (frequencies / head)	16	16
λ (kernel decay)	2.0	2.0
RPE warmup epochs	50 (cosine)	30 (cosine)
Evaluation		
Crop size	480×640	480×480
Scales	$\{1.0\}$	$\{1.0\}$
Horizontal flip	✓	✓
Precision		
Training	FP32	FP32
Validation	AMP	AMP

Camera Intrinsics. NYU Depth v2 uses a single Microsoft Kinect v1 sensor (f_x=518.86, f_y=519.47, c_x=325.58, c_y=253.74); all images share identical intrinsics.

SUN RGB-D aggregates data from four sensors with varying focal lengths:

Per-image intrinsics are loaded from sensor metadata (98.8% coverage; unmatched images default to Kinect v2 values). Pixels are back-projected to 3D world coordinates and normalized per-image to zero mean and unit variance before computing the action of the mask matrix.

Table 9: **Sensor distribution** in SUN RGB-D.

Sensor	Images	f_x	f_y
Kinect v1	1,993	518.86	519.47
Kinect v2	3,684	529.5	529.5
Asus Xtion	3,384	570.3	570.3
RealSense	1,152	693.7	693.7

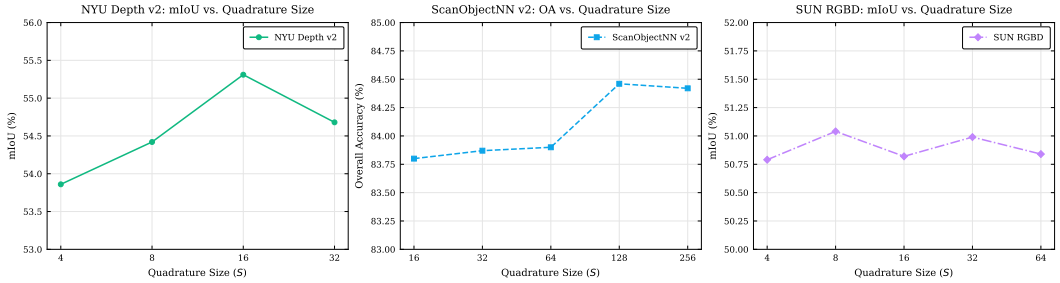


Figure 5: Effect of the quadrature size S per head across different datasets. The plots illustrate the performance trade-offs on NYU Depth v2 (left, mIoU%), ScanObjectNN v2 (middle, Overall Accuracy%), and SUN RGB-D (right, mIoU%) for $\lambda = 2$.

RPE Warmup Schedule. On NYU, applying the full RPE mask from epoch 0 yields no improvement over the Performer baseline (54.42 vs. 54.44 mIoU). We attribute this to the RPE mask interfering with the pretrained backbone during early optimization. A warmup schedule resolves this: the RPE mask M is blended from identity to full strength over the first N epochs via

$$\hat{M} = \mathbf{1} + \alpha\,(M - \mathbf{1}), \qquad \alpha = \tfrac{1}{2}\big(1 - \cos(\pi\ \min(t/N,\ 1))\big), \tag{16}$$

where t is the current epoch. At α=0 the model reduces to a pure Performer; at α=1 the full RPE is applied. With cosine warmup (N=50 on NYU, N=30 on SUN), the result improves to **55.32** mIoU on NYU (+0.88 over Performer-only) and **50.65** mIoU on SUN (+2.16 over Performer-only).

All experiments run on NVIDIA RTX PRO 6000 (Blackwell) GPUs with 98 GB VRAM. We use FP32 training throughout.

C Ablation Studies

In this section we will study the effect of the quadrature size S and the modulation parameter λ.

C.1 Quadrature Size

We ablate the quadrature size S across various datasets. A small quadrature suffices for optimal performance.

As shown in Fig. 5, increasing S improves performance only up to a task-dependent saturation point, with λ fixed to 2.0. On NYU Depth v2, the best result is achieved with S=16, while ScanObjectNN reaches its peak accuracy at S=128 with almost no further gain from S=256.

Finally, we show that the quadrature size remains constant even when higher number of points are sampled per point cloud. Table 10 shows that higher quadrature size is not necessary even when 2048 points are sampled per point cloud.

Table 10: **Effect of the size of the quadrature** S per head on ScanObjectNN v2 (Overall Accuracy %). λ=2.

	S=16	S=32	S=64	S=128	S=256
OA	83.12	83.54	83.77	**84.26**	84.12

C.2 Modulation parameter

In this section we ablate over λ across various datasets.

Table 11: Ablation of the modulation parameter λ on indoor (ScanNet, ScanNet200, and ScanNet++) and outdoor (nuScenes) semantic segmentation benchmarks.

Methods	ScanNet Val [15]			ScanNet200 Val [62]			ScanNet++ Val [96]			nuScenes Val [5]		
	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
Transformer	77.6	85.0	92.0	35.3	46.0	83.4	48.2	61.6	87.0	80.4	87.2	94.7
Performer	74.8	83.8	91.0	28.2	39.6	80.2	48.1	62.5	87.2	72.0	80.2	93.5
RelFlexformer ($\lambda=1.0$)	76.9	84.8	91.8	34.0	45.4	82.9	48.7	62.2	86.8	79.9	87.3	94.5
RelFlexformer ($\lambda=2.0$)	76.8	85.0	91.9	33.9	44.4	82.7	48.3	62.4	87.1	80.0	87.3	94.5
RelFlexformer ($\lambda=3.0$)	75.7	84.1	91.5	33.8	45.2	82.7	48.7	62.0	87.0	80.3	87.5	94.6

Table 12: Ablation of the modulation parameter λ on S3DIS 6-fold cross-validation.

Method	**Metric**	**Area1**	**Area2**	**Area3**	**Area4**	**Area5**	**Area6**	**6-Fold**
Transformer	allAcc	93.22	86.26	94.56	90.72	91.67	94.98	91.90
	mAcc	89.92	74.44	94.45	81.11	78.92	93.55	85.31
	mIoU	83.01	63.42	86.66	71.34	73.43	87.31	77.70
Performer	allAcc	92.35	88.53	94.47	85.96	90.86	91.20	90.56
	mAcc	88.56	76.97	93.69	78.31	75.63	85.77	83.16
	mIoU	80.74	62.60	86.02	62.60	69.75	77.35	73.18
RelFlexformer ($\lambda=1.0$)	allAcc	92.96	88.07	94.51	88.36	91.01	94.75	91.61
	mAcc	90.59	77.33	93.00	80.92	75.95	93.15	85.16
	mIoU	81.93	63.21	86.20	68.28	70.60	86.58	76.13
RelFlexformer ($\lambda=2.0$)	allAcc	92.88	86.69	93.84	88.16	91.08	94.65	91.22
	mAcc	89.67	74.53	90.95	79.58	77.15	93.33	84.20
	mIoU	81.58	62.82	83.44	67.55	71.13	86.88	75.57
RelFlexformer ($\lambda=3.0$)	allAcc	92.72	88.43	93.88	88.03	91.14	94.48	91.45
	mAcc	89.26	77.35	92.69	79.79	76.79	92.70	84.76
	mIoU	81.48	62.76	84.64	66.45	71.01	86.16	75.42

We study the effect of the modulation parameter λ for the exponential decay by evaluating $\lambda \in \{1.0, 2.0, 3.0\}$.

Segmentation on Point Clouds RelFlexformer outperforms Performer across all tested values of λ on the main benchmark-level mIoU metrics, indicating that the proposed modulation does not rely on a single carefully tuned setting. Overall, $\lambda = 1.0$ gives the best mIoU on ScanNet, ScanNet200, and the S3DIS 6-fold evaluation, while $\lambda = 3.0$ performs best on nuScenes and ties with $\lambda = 1.0$ on ScanNet++. For S3DIS Area 5, $\lambda = 2.0$ achieves the best mIoU. This suggests that different scene distributions may benefit from different spatial modulation scales. Nevertheless, the performance variation across λ is relatively small compared with the improvement over Performer, showing that RelFlexformer provides robust gains under different modulation strengths.

While dense PTv3 attention remains a strong reference, RelFlexformer consistently narrows the gap between Performer and Transformer attention and even matches or exceeds the dense-attention reference in some metrics. These results support our main claim that Performer-style efficient attention benefits from explicit geometric modulation when applied to irregular 3D data.

Classification and RGBD data We ablate over λ across all three benchmarks (Table 13). $\lambda = 2$ is optimal or near-optimal in these settings, improving over the performer baseline.

Table 13: Effect of λ across datasets. Best result per dataset in **bold**.

λ	NYU (mIoU)	SUN (mIoU)	ScanObjNN (OA)
1	54.34	48.56	83.24
2	**55.32**	**51.04**	**84.45**
3	54.08	48.18	82.79
Performer	54.44	48.49	83.17

D Additional Comparisons

In this section, we provide additional benchmark comparisons for the results reported in Tables 2 and 3. While the main text focuses on controlled evaluations within the same PTv3 framework, the following tables compare RelFlexformer with representative prior methods reported on the corresponding datasets. These comparisons are intended to place RelFlexformer in a broader empirical context across commonly reported point-based, convolutional, Transformer-based, Mamba-based, and Performer-based methods.

D.1 Classification

Table 14: Classification accuracy on ModelNet40 using 1,024 input points without voting. Bold denotes the best result in the Performer group.

Method	OA
CNN	
PointNet [56]	89.2
A-SCN [88]	89.9
PointNet++ [57]	90.7
PointGrid [36]	92.0
PCNN [2]	92.3
PointCNN [38]	92.5
PointConv [83]	92.5
P2Sequence [44]	92.6
DGCNN [81]	92.9
RS-CNN [45]	92.9
PointASNL [92]	92.9
Transformer	
PCT [23]	93.2
OctFormer [77]	92.7
Mamba	
Point Mamba (w/ OctFormer) [43]	92.7
Point Mamba (w/ PCT) [43]	93.4
PointMamba [40]	93.6
PCM [105]	93.4
Performer	
Performer (w/ PCT)	92.3
RelFlexformer (w/ PCT) (ours)	**92.9**
+ PointRoPE [101]	92.6

Table 15: Classification accuracy on ScanObjectNN v2 using 1,024 input points, except PointMAE and PointMamba using 2,048 points, without voting. Bold denotes the best result in the Performer group.

Method	OA
MLP/CNN	
PointNet [56]	68.2
PointNet++ [57]	77.9
SpiderCNN [90]	73.7
PointCNN [38]	78.5
DGCNN [81]	78.1
PointMLP [50]	85.7
PointNeXt [58]	87.7
Transformer	
Point-BERT [100]	83.1
MaskPoint [42]	84.3
Point-MAE [51]	85.2
Point-M2AE [104]	86.4
PCT [23]	84.0
Mamba	
PointMamba [40]	84.9
PCM-Tiny [105]	86.9
Performer	
Performer (w/ PCT)	83.2
RelFlexformer (w/ PCT) (ours)	**84.5**
+ PointRoPE [101]	84.3

D.2 Segmentation

Table 16: Semantic segmentation results on ScanNet v2. Val denotes validation mIoU. Bold denotes the best result in the Performer group.

Method	Val
MLP/CNN	
PointNet++ [57]	53.5
SparseConvNet [21]	69.3
PointConv [83]	61.0
JointPointBased [10]	69.2
KPConv [73]	69.2
PointASNL [92]	63.5
PointNeXt [58]	71.5
LargeKernel3D [9]	73.5
PointMetaBase [41]	72.8
MinkowskiNet [14]	72.2
Transformer	
Fast Point Transformer [52]	72.1
Stratified Transformer [34]	74.3
PointConvFormer [82]	74.5
OctFormer [77]	75.7
Swin3D [95]	76.4
PTv1 [107]	70.6
PTv2 [85]	75.4
PTv3 [84]	77.5
Mamba	
Point Mamba [43]	74.6
Pamba [39]	77.6
Performer	
Performer (w/ PTv3)	74.8
RelFlexformer (w/ PTv3) (ours)	**76.9**
+ PointRoPE [101]	76.6

Table 17: Semantic segmentation results on S3DIS (Area 5). Scores denote mIoU on Area 5. Bold denotes the best result in the Performer group.

Method	Area 5
MLP/CNN	
PointNet [56]	41.1
SegCloud [72]	48.9
TanConv [71]	52.6
PointCNN [38]	57.3
ParamConv [78]	58.3
PointWeb [106]	60.3
HPEIN [29]	61.9
KPConv [73]	67.1
GACNet [76]	62.9
SPGraph [35]	58.0
SegGCN [37]	63.6
PAConv [89]	66.6
PointNeXt [58]	70.5
PointMetaBase [41]	72.0
MinkUNet [14]	65.4
Transformer	
PAT [94]	60.1
Stratified Transformer [34]	72.0
Superpoint Transformer [61]	68.9
Swin3D [95]	72.5
PTv1 [107]	70.4
PTv2 [85]	71.6
PTv3 [84]	73.4
Mamba	
Pamba [39]	73.5
Performer	
Performer (w/ PTv3)	69.8
RelFlexformer (w/ PTv3) (ours)	71.0
+ PointRoPE [101]	**72.1**

Table 18: Semantic segmentation results on nuScenes. Val denotes validation mIoU. Bold denotes the best result in the Performer group.

Method	**Val**
MLP/CNN/SparseConv	
SPVNAS [70]	77.4
Cylinder3D [110]	76.1
PVKD [26]	76.0
2DPASS [91]	79.4
MinkUNet [14]	73.3
+ M3Net [46]	79.0
+ PPT [86]	78.6
OA-CNNs [53]	78.9
Transformer	
SphereFormer [33]	78.4
RangeFormer [32]	78.1
PTv2 [85]	80.2
PTv3 [84]	80.4
Mamba	
Pamba [39]	80.4
Performer	
Performer (w/ PTv3)	72.0
RelFlexformer (w/ PTv3) (ours)	80.3
+ PointRoPE [101]	**81.2**

D.3 Segmentation on RGB-D datasets

Table 19: Segmentation results on NYU Depth v2. Bold denotes the best result in the Performer group.

Method	Backbone	Params	mIoU
ACNet [27]	ResNet-50	116.6M	48.3
SGNet [7]	ResNet-101	64.7M	51.1
SA-Gate [8]	ResNet-101	110.9M	52.4
CEN [80]	ResNet-101	118.2M	51.7
CEN [80]	ResNet-152	133.9M	52.5
ShapeConv [6]	ResNext-101	86.8M	51.3
ESANet [65]	ResNet-34	31.2M	50.3
FRNet [109]	ResNet-34	85.5M	53.6
PGDENet [108]	ResNet-34	100.7M	53.7
EMSANet [64]	ResNet-34	46.9M	51.0
TokenFusion [79]	MiT-B2	26.0M	53.3
TokenFusion [79]	MiT-B3	45.9M	54.2
MultiMAE [3]	ViT-B	95.2M	56.0
Omnivore [20]	Swin-Tiny	29.1M	49.7
Omnivore [20]	Swin-Small	51.3M	52.7
Omnivore [20]	Swin-Base	95.7M	54.0
CMX [103]	MiT-B2	66.6M	54.4
DFormer [99]	DFormer-Tiny	6.0M	51.8
DFormer [99]	DFormer-Small	18.7M	53.6
DFormer [99]	DFormer-Base	29.5M	55.6
AsymFormer [18]	MiT-B0+ConvNeXt-Tiny	33.0M	55.3
Performer (w/ DFormer)	DFormer-Base	29.5M	54.4
RelFlexformer (w/ DFormer) (ours)	DFormer-Base	29.5M	**55.3**

Table 20: Segmentation results on SUN RGB-D. Bold denotes the best result in the Performer group.

Method	Backbone	Params	mIoU
ACNet [27]	ResNet-50	116.6M	48.1
SGNet [7]	ResNet-101	64.7M	48.6
SA-Gate [8]	ResNet-101	110.9M	49.4
CEN [80]	ResNet-101	118.2M	50.2
CEN [80]	ResNet-152	133.9M	51.1
ShapeConv [6]	ResNet-101	86.8M	48.6
ESANet [65]	ResNet-34	31.2M	48.2
FRNet [109]	ResNet-34	85.5M	51.8
PGDENet [108]	ResNet-34	100.7M	51.0
EMSANet [64]	ResNet-34	46.9M	48.4
TokenFusion [79]	MiT-B2	26.0M	50.3
TokenFusion [79]	MiT-B3	45.9M	51.0
MultiMAE [3]	ViT-B	95.2M	51.1
CMX [103]	MiT-B2	66.6M	49.7
DFormer [99]	DFormer-Tiny	6.0M	48.8
DFormer [99]	DFormer-Small	18.7M	50.0
DFormer [99]	DFormer-Base	29.5M	51.2
AsymFormer [18]	MiT-B0+ConvNeXt-Tiny	33.0M	49.1
Performer (w/ DFormer)	DFormer-Base	29.5M	48.5
RelFlexformer (w/ DFormer) (ours)	DFormer-Base	29.5M	**51.0**

E Computational Efficiency and Memory Footprint

To evaluate the operational scalability and efficiency of RelFlexformer, we benchmark peak memory consumption, computational throughput (FLOPs), and execution latency against standard Softmax attention and Performer [12]. Benchmarks are conducted on two representative modern architectures: Point Transformer v3 (PTv3) (46.2M parameters) for large-scale 3D point cloud perception, and

DFormer for fine-grained spatial-geometry tasks. All physical runtime measurements were performed on a single NVIDIA RTX PRO 6000 GPU (96 GB VRAM).

E.1 PTv3 Runtime and Scalability Analysis

We first analyze the training and inference behavior of the PTv3 encoder as a function of the patch size K. Standard projection and feed-forward network (FFN) overheads remain invariant across attention mechanisms; hence, these measurements isolate the core efficiency gains provided by our operator.

Table 21: **PTv3 full encoder benchmark.** Peak memory footprint and per-step execution latency across varying patch sizes K on an NVIDIA RTX PRO 6000 (96 GB). Latency is measured in milliseconds (ms), and memory in gigabytes (GB). OOM denotes Out-of-Memory (> 96 GB).

	PTv3 Encoder Performance Across Patch Sizes (K)					
	Softmax		Performer (no RPE)		**RelFlexformer (Ours)**	
Patch Size (K)	Memory	Latency	Memory	Latency	Memory	Latency
(a) Full Encoder Training Step (Forward + Backward)						
1024 (Default)	75.0 GB	713 ms	0.9 GB	19 ms	**4.4 GB**	**52 ms**
2048	OOM		1.1 GB	19 ms	5.6 GB	51 ms
4096	OOM		1.5 GB	21 ms	8.4 GB	75 ms
(b) Encoder Inference (Forward-Only)						
1024 (Default)	32.0 GB	91.2 ms	0.2 GB	0.2 ms	**1.1 GB**	**8.4 ms**
2048	63.9 GB	173.8 ms	0.2 GB	0.2 ms	1.1 GB	8.3 ms
4096	OOM		0.2 GB	0.2 ms	1.3 GB	8.0 ms
16384	OOM		0.2 GB	0.3 ms	1.6 GB	8.2 ms

As summarized in Table 21, standard Softmax attention exhibits quadratic memory complexity, suffering from Out-of-Memory (OOM) failures at $K \geq 2048$ during training and $K \geq 4096$ during inference. While Performer offers ultra-low overhead, it lacks relative positional encoding capabilities essential for structural precision. RelFlexformer bridges this gap: it cuts peak training memory at $K = 1024$ from 75.0 GB to **4.4 GB** (a $17\times$ reduction) while reducing step latency from 713 ms to **52 ms** ($13.7\times$ speedup), enabling training at ultra-large context lengths ($K = 4096$) with modest hardware requirements.

E.2 Theoretical Computational Complexity (GFLOPs)

To further evaluate computational scaling independent of hardware hardware-specific kernel optimizations, we isolate the total attention-core GFLOPs across fixed point-cloud configurations ($N = 65,536$ points) and multi-stage vision backbones.

Table 22: **Attention-core computational cost (GFLOPs).** Comparison isolating the core attention operation across varying patch scales and architecture stages.

Configuration / Stage	**Tokens (N)**	**Softmax**	**Performer**	**RelFlexformer (Ours)**
(a) PTv3 Attention Core ($N = 65,536$ fixed points, patch size K varied)				
$K = 1024$ (Default)	65,536	43.5	0.55	**18.0**
$K = 4096$	65,536	174.0	0.55	**18.0**
$K = 16384$	65,536	696.0	0.55	**18.0**
(b) DFormer Backbone Stage-wise Dense Attention ($N_q = N_k = H \times W$)				
Stage 1 (120×160)	19,200	100.6	0.32	**10.4**
Stage 2 (60×80)	4,800	12.6	0.16	**5.2**
Stage 3 (30×40)	1,200	1.6	0.08	**2.6**

As demonstrated in Table 22(a), as the patch size K scales to 16,384, Softmax attention cost explodes to 696.0 GFLOPs. In contrast, RelFlexformer maintains a constant **18.0 GFLOPs**, achieving linear complexity with respect to input scale.

Furthermore, Table 22(b) highlights the impact on dense spatial models like DFormer. At its native Stage 1 resolution ($120 \times 160 = 19,200$ tokens), full global Softmax attention requires **100.6 GFLOPs** and **94.6 GB** VRAM, rendering standard training infeasible. Consequently, existing models resort to spatial pooling heuristics to reduce query counts artificially. RelFlexformer reduces Stage 1 computation to **10.4 GFLOPs** and peak memory to **2.9 GB** (a $\sim 10\times$ decrease in compute), entirely removing the need for query-downsampling constraints while preserving full-resolution global geometry.

F Broader Impact

RelFlexFormer enables linear-time attention with relative position encoding for 3D perception (RGB-D segmentation, point cloud classification and segmentation). Practical applications include robotics (real-time scene understanding), autonomous navigation, and AR/VR — domains where quadratic attention is a latency bottleneck on edge devices. The method is a drop-in replacement for softmax attention in existing architectures, lowering the compute barrier for deploying transformer-based 3D perception. We do not foresee negative societal impacts beyond those inherent to improved scene understanding (e.g., surveillance), which are shared with all methods in this space.

G Limitations

We use $\exp(-\lambda|\mathbf{x}|)$ as our $\mathcal{F}$ throughout our work. While our work is general and can potentially use any L_1 function, the search for the function per domain remains an open question. Finally, our work only focuses on functions that have a well-defined Fourier transforms, thus extending it to any smooth function remains a challenge.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope, including RelFlexformer, its RPE-based modulation mechanism, and its fast subquadratic matrix-vector multiplication. All claims are supported by the content.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We clearly discuss the limitations of RelFlexformer. We include a limitations section in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide the theorems, definitions, and formulas for RelFlexformer in the main paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose all necessary details to reproduce our main results, including architectural and configurations in the main paper (Section 4) and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We use publicly available datasets and build on publicly available baseline code. We provide all hyperparameters and implementation details needed to reproduce our experiments in Appendix B. Code for our method is available at `https://github.com/97byeongchan/RelFlexformer`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide all hyperparameters and implementation details needed to reproduce our experiments in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report standard deviation for experiments to provide a measure of statistical significance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We disclose the compute resources used for our experiments, including GPU type, and memory.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper fully adheres to the NeurIPS Code of Ethics. We have carefully reviewed the ethical guidelines and ensured that our research complies with them in all respects.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have included the broader impact section in Appendix F.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any model or dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available datasets and code as published in the original papers, and we cite all the original sources.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not involve any crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not involve any crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We did not use LLMs in our work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.