SAMPLED TRANSFORMER FOR POINT SETS

Anonymous authors
Paper under double-blind review

ABSTRACT

The sparse transformer can reduce the computational complexity of the self-attention layers to $O(n)$, whilst still being a universal approximator of continuous sequence-to-sequence functions. However, this permutation variant operation is not appropriate for direct application to sets. In this paper, we proposed an $O(n)$ complexity sampled transformer that can process point set elements directly without any additional inductive bias. Our sampled transformer introduces random element sampling, which randomly splits point sets into subsets, followed by applying a shared Hamiltonian self-attention mechanism to each subset. The overall attention mechanism can be viewed as a Hamiltonian cycle in the complete attention graph, and the permutation of point set elements is equivalent to randomly sampling Hamiltonian cycles. This mechanism implements a Monte Carlo simulation of the $O(n^2)$ dense attention connections. We show that it is a universal approximator for continuous set-to-set functions. Experimental results for classification and few-shot learning on point-clouds show comparable or better accuracy with significantly reduced computational complexity compared to the dense transformer or alternative sparse attention schemes.

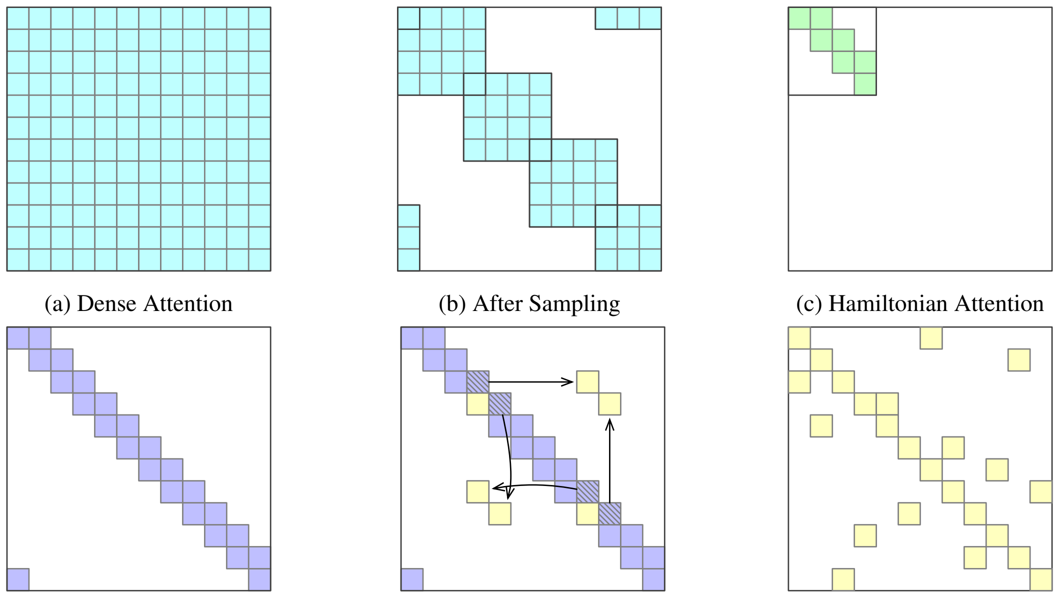
1 INTRODUCTION

Encoding structured data has become a focal point of modern machine learning. In recent years, the defacto choice has been to use transformer architectures for sequence data, *e.g.*, in language (Vaswani et al., 2017) and image (Dosovitskiy et al., 2020) processing pipelines. Indeed, transformers have not only shown strong empirical results, but also have been proven to be universal approximators for sequence-to-sequence functions (Yun et al., 2019). Although the standard transformer is a natural choice for set data, with permutation invariant dense attention, its versatility is limited by the costly $O(n^2)$ computational complexity. To decrease the cost, a common trick is to use sparse attention, reducing the complexity from $O(n^2)$ to $O(n)$ (Yun et al., 2020; Zaheer et al., 2020; Guo et al., 2019). However, in general this results in an attention mechanism that is not permutation invariant – swapping two set elements change which elements they attend. As a result, sparse attention cannot be directly used for set data.

Recent work has explored the representation power of transformers in point sets as a plug-in module (Lee et al., 2019), a pretraining-finetuning pipeline (Yu et al., 2022; Pang et al., 2022), and with a hierarchical structure (Zhao et al., 2021). However, these set transformers introduced additional inductive biases to (theoretically) approach the same performance as the densely connected case in language and image processing applications. For example, to achieve permutation invariance with efficient computational complexity, previous work has required nearest neighbor search (Zhao et al., 2021) or inducing points sampling (Lee et al., 2019). Following the above analysis, a research question naturally arises to avoid introducing unneeded inductive bias:

Can $O(n)$ complexity sparse attention mechanisms be applied directly to sets?

We propose the sampled transformer to address this question, which is distinguished from the original sparse transformer by mapping the permutation of set elements to the permutation of attention matrix elements. Viewing this permutation sampling as attention matrix sampling, the proposed sampled attention approximates $O(n^2)$ dense attention. This is achieved with the proposed random element sampling and Hamiltonian self-attention. To be specific, in random element sampling the input point set is first randomly split into several subsets of n_s points (Fig. 1b), each of which will be processed by shared self-attention layers. In addition, a sparse attention mechanism – namely



(a) Dense Attention (b) After Sampling (c) Hamiltonian Attention

(d) Cycle Attention (e) Swap Points (f) Sampled Attention

Figure 1: Attention mechanisms: *(a)* original *dense attention*; *(b)* the attention matrix after random element sampling; *(c)* a special case of sparse attention – *Hamiltonian (self-)attention* – for each subset; *(d)* combining all subsets (which have overlapping element per (b)) connects the individual Hamiltonian attention sub-matrices, gives *cycle attention* which is a Hamiltonian cycle; *(e)* permutation of points permutes the elements in cycle attention matrix; *(f)* the resulting *sampled attention*, viewed as a sampled Hamiltonian cycle from the edges of the complete attention graph.

Hamiltonian self-attention (Fig. 1c) – is applied to reduce complexity of the subset inputs, so that n_s point connections are sampled from $O(n_s^2)$ connections. The combination of all Hamiltonian self-attention mechanism for all subsets – namely *cycle attention* (Fig. 1d) – can be viewed as a Hamiltonian cycle in the complete attention graph. As a result, the permutation of set elements is equivalent to the permutation of nodes in a Hamiltonian cycle (Fig. 1e), which is in fact randomly sampling Hamiltonian cycles from the complete graph – thereby yielding the proposed *sampled attention* (Fig. 1f). Finally, viewing this randomization as a Monte Carlo sample of attention pairs, repeated sampling can be used to approximate the complete $O(n^2)$ dense connections. Furthermore, our proposed sampled transformer is proven to be a universal approximator for set data – any continuous set-to-set functions can be approximated to arbitrary precision.

The contributions of this paper are summarized as follows.

- We propose the sampled attention mechanism which maps the random permutation of set elements to the random sampling of Hamiltonian cycle attention matrices, permitting the direct processing of point sets.
- We prove that the proposed sampled transformer is a universal approximator of continuous set-to-set functions, see Corollary 1.
- Compared to previous transformer architectures, the empirical results show that our proposed sampled transformer achieves comparable (or better) performance with less inductive bias and complexity.

2 Related Work

The transformer (Vaswani et al., 2017) is widely used in languages (Raffel et al., 2020; Dai et al., 2019; Yang et al., 2019b) and images (Dosovitskiy et al., 2020; Liu et al., 2021; Touvron et al., 2021; Ramachandran et al., 2019). For example, Raffel et al. (2020) explored the transformer by unifying a suite of text problems to a text-to-text format; Dai et al. (2019) modeled very long-term dependency by reusing previous hidden states; Dosovitskiy et al. (2020) demonstrated that the pure transformer can be effectively applied directly to a sequence of image patches; and Liu et al. (2021) proposed a transformer with hierarchical structure to learn various scales with linear computational complexity. In addition, the representation power of the transformer has been explored by the pre-training and fine-tuning models (Devlin et al., 2018; Bao et al., 2021; Yu et al., 2022; He et al., 2022).

Recently, an increasing number of researchers begin to explore the representation power of the transformer in 3D point clouds (sets) data. Xie et al. (2018) applied multi-layered dense transformers to small-scale point clouds directly; Yang et al. (2019a) further proposed the Group Shuffle attention to deal with size-varying inputs by furthest point sampling; Han et al. (2022) aggregated point-wise and channel-wise features by directly adding two self-attention layers. To avoid the tricky tokenization step, Lee et al. (2019) tried to deal with points directly with $O(nm)$ complexity by introducing inducing points, and proved universal approximation; Mazur & Lempitsky (2021) further proposed a hierarchical point set mapping, grouping, and merging structure with nearest neighbors defining the sparse attention mechanism. Yu et al. (2022) and Pang et al. (2022) further introduced the transformers to the pre-training and fine-tuning pipelines in the area of 3D point clouds. Last but not the least, transformers have also been widely used in other such works on 3D (point cloud) data as Liu et al. (2019a); Misra et al. (2021); Mao et al. (2021); Fuchs et al. (2020); Sander et al. (2022)

Another important line of work seeks to theoretically demonstrate the representation power of the transformer by showing the universal approximation of continuous sequence-to-sequence functions (Yun et al., 2019; 2020; Zaheer et al., 2020; Shi et al., 2021; Kratsios et al., 2021). To be specific, Yun et al. (2019) demonstrated the universal approximation property of the transformer; Yun et al. (2020) and Zaheer et al. (2020) demonstrated that the transformer with sparse attention matrix remains a universal approximator; Shi et al. (2021) claimed that the transformer without diag-attention is still a universal approximator. Kratsios et al. (2021) proposed that the universal approximation under constraints is possible for the transformer.

In comparison with the above works, we proposes the $O(n)$ sampled transformer – a universal approximator of continuous set-to-set functions. To our knowledge, the use of approximating dense attention by sampling Hamiltonian cycle attention matrices is new.

3 Preliminary

3.1 Notation

Given an integer a we define $[a] \doteq \{1, \dots, a\}$. For a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times m}$, for a $k \in [m]$ the k-th column is denoted by $\boldsymbol{M}_k$. Given an (ordered) index set $\mathcal{A} \subset [m]$ the submatrix $\boldsymbol{M}_\mathcal{A} \in \mathbb{R}^{n\times|\mathcal{A}|}$ consists of the matrix generated by concatenating the columns determined by indices in $\mathcal{A}$. See the notation guide in §A in the supplementary material.

3.2 Transformer

The transformer $\boldsymbol{X} \mapsto t(\boldsymbol{X})$ (Vaswani et al., 2017; Dosovitskiy et al., 2020) implements a function from point clouds to point clouds with input points $\boldsymbol{X} \in \mathbb{R}^{d\times n}$. It is formally defined by a multi-head self-attention layer and a feed-forward layer:

$$\text{Head}^j(\boldsymbol{X}) = (\boldsymbol{W}_V^j\boldsymbol{X}) \cdot \sigma_S[(\boldsymbol{W}_K^j\boldsymbol{X})^T\boldsymbol{W}_Q^j\boldsymbol{X}] \tag{1a}$$

$$\text{Attn}(\boldsymbol{X}) = \boldsymbol{X} + \boldsymbol{W}_O \begin{bmatrix}\text{Head}^1(\boldsymbol{X})\\ \vdots \\ \text{Head}^h(\boldsymbol{X})\end{bmatrix} \tag{1b}$$

$$\text{TB}(\boldsymbol{X}) = \text{Attn}(\boldsymbol{X}) + \boldsymbol{W}_2 \cdot \text{ReLU}(\boldsymbol{W}_1\text{Attn}(\boldsymbol{X})), \tag{1c}$$

where n is the number of points and d is the feature dimension. Head$(\cdot)$ is the **self-attention layer**, and Attn$(\cdot)$ is the **multi-head self-attention layer** with the parameter $\boldsymbol{W}_O \in \mathbb{R}^{d\times mh}$. $\boldsymbol{W}_V^i, \boldsymbol{W}_K^i, \boldsymbol{W}_Q^i \in \mathbb{R}^{m\times d}$ are value, key, and query parameters; $\boldsymbol{W}_1 \in \mathbb{R}^{r\times d}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d\times r}$ are feed-forward layer parameters. We utilize a positional embedding $\boldsymbol{E}$ in the input $\boldsymbol{X}$, defined by $\boldsymbol{E} = \boldsymbol{W}_p\boldsymbol{P}$, where $\boldsymbol{P} \in \mathbb{R}^{3\times n}$ is the (xyz) coordinate, and $\boldsymbol{W}_P \in \mathbb{R}^{d\times 3}$ is an MLP layer. To simplify the notation, here we use $\boldsymbol{X} = \boldsymbol{X} + \boldsymbol{E}$ so that all the inputs $\boldsymbol{X}$ in this paper will include the positional embedding unless specifically stated otherwise. The **attention mechanism** for a dense transformer is the $n \times n$ attention matrix $(\boldsymbol{W}_K^i \boldsymbol{X})^T \boldsymbol{W}_Q^i \boldsymbol{X}$ in Eq. 1a, which is in fact a similarity matrix for n elements/tokens, or a **complete attention graph**. **Sparse Attention** also refers to the same similarity matrix/attention graph but with sparse connections instead. As tokenization may not be necessary in dealing with point clouds, for clarity we use the terminology points, elements, and tokens are all to refer to points (which may be thought of as tokens in a traditional transformer context) in a point cloud (set).

3.3 Universal Approximation

Let $\mathcal{F}$ be the class of continuous sequence-to-sequence functions $f : \mathbb{R}^{d\times n} \mapsto \mathbb{R}^{d\times n}$ defined on any compact domain. Further define $\mathcal{T}^{h,m,r}$ as the set of transformer blocks $t(\cdot)$ with h attention heads of each of size m, and with hidden layer width r (Yun et al., 2019; 2020). To measure the distance between functions in $\mathcal{F}$, we define the standard ℓ_p distance function by the corresponding norm:

$$d_p(f_1, f_2) = \left(\int \|f_1(\boldsymbol{X}) - f_2(\boldsymbol{X})\|_p^p \, \mathrm{d}\boldsymbol{X} \right)^{1/p}, \tag{2}$$

which is element-wise continuous (w.r.t the ℓ_p norm) for $1 \leq p < \infty$.

Theorem 1 (Universal Approximation, Yun et al. (2019))**.** *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}$, there exist a Transformer network $g \in \mathcal{T}^{2,1,4}$, such that $d_p(f, g) \leq \epsilon$.*

The proof of Theorem 1 makes three stages of approximations, which are chained together via the triangle inequality to give the ϵ bound (Yun et al., 2019). In particular, ① any $f \in \mathcal{F}$ is approximated by a piece-wise linear function $\overline{f} \in \overline{\mathcal{F}}$ (over a discretized input space). Then ② the piece-wise linear function is approximated by a *modified* transformer $\overline{\mathcal{T}}^{2,1,4}$, where the widely used ReLU and σ_S activation functions (as per Eq. 1) are replaced by the hardmax function σ_H. Finally, ③ it is shown that the class of transformer $\overline{\mathcal{T}}^{2,1,4}$ can approximate any regular transformer $g \in \mathcal{T}^{2,1,4}$.

The key step comes in the proof of the second approximation ②. In Yun et al. (2019), the approximation is proved by showing that multi-head self-attention layers of the modified transformer can implement any contextual map $q_c : \mathbb{R}^{d\times n} \mapsto \mathbb{R}^n$.

Definition 3.1 (Contextual Mapping)**.** *Consider a finite set $\mathbb{L} \subset \mathbb{R}^{d\times n}$. A map q: $\mathbb{L} \mapsto \mathbb{R}^{1\times n}$ defines a contextual map if the map satisfies the following:*

1. *For any $\boldsymbol{L} \in \mathbb{L}$, the n entries in $q(\boldsymbol{L})$ are all distinct.*
2. *For any $\boldsymbol{L}, \boldsymbol{L}' \in \mathbb{L}$, with $\boldsymbol{L} \neq \boldsymbol{L}'$, all entries of $q(\boldsymbol{L})$ and $q(\boldsymbol{L}')$ are distinct.*

Intuitively, a contextual map can be thought of as a function that outputs unique "id-values". The only way for a token (column) in $\boldsymbol{L} \subset \mathbb{R}^{d\times n}$ to share an "id-value" (element of $q(\boldsymbol{L})$) is to map the exact same sequence. As each token in the sequence is mapped to a unique value, an appropriately constructed feed-forward neural network can map a sequence to any other desired sequence, providing a universal approximation guarantee. In Yun et al. (2020), such a contextual map is implemented via *selective shift operators* and *all-max-shift operators* through careful construction of multi-head self-attention layers.

4 Methodology

We propose a variation of the sparse attention transformer – sampled sparse attention transformer – applicable to point sets. We deviate from the typical sparse attention transformer in two ways. First, we randomly sub-sample the input point set l times, with each sub-sample being evaluated through a shared multi-head self-attention layer. Secondly, we propose a simple Hamiltonian self-attention mechanism, a special case of the sparse attention mechanism, to reduce the computation complexity of considering point sets. This ultimately yields the variant of the typical sparse transformer (Eq. 1) which can be interpreted as using a *sampled attention* mechanism, as depicted in Fig. 1. To study the approximation capabilities of our proposed architecture, we prove that our sampled sparse attention transformer is a universal approximator of set-to-set functions.

4.1 Random Element Sampling

For a point set input $\boldsymbol{X} \in \mathbb{R}^{d \times n}$, instead of directly applying the transformer attention layer to n tokens, we process l many sub-sampled inputs $\mathbf{X}^i \in \mathbb{R}^{d \times n_s}$ for $i \in [l]$ and $2 \leq n_s \leq n$. For simplicity, we assume that $(n_s - 1) \cdot l = n$. The sub-sampled inputs $\mathbf{X}^i$ can be defined by taking various column submatrices:

$$\mathbf{X}^i = \boldsymbol{X}_{\mathcal{R}^i \cup \mathcal{R}_1^{\gamma(i)}}; \quad \gamma(v) = 1 + (v \mod l), \tag{3}$$

where $\mathcal{R}^1, \ldots \mathcal{R}^l$ are randomly selected ordered index sets, such that $|\mathcal{R}^i| = n_s - 1$ and $\mathcal{R}^i \cap \mathcal{R}^j = \emptyset$ for $i \neq j$. The index element $\mathcal{R}_1^i$ denotes the first index in the ordered set $\mathcal{R}^i$. The *cycle* function $\gamma : [l] \to [l]$ ensures that the edge-case of $\mathbf{X}^l$ is well defined, *i.e.*, $\gamma(l) = 1$.

Intuitively, the sequence of sub-sampled inputs $\mathbf{X}^1, \ldots, \mathbf{X}^l$ can be interpreted as a rolling window of $(n_s - 1) \cdot l = n$ many sampled point set elements. Indeed, by concatenating the index sets in order, $\mathbf{X}^i$ is a sliding window of the elements with size n_s and stride $n_s - 1$ (with wrapping).

It should be noted that $\mathbf{X}^i$ can be treated as a random variable. As such a singular realization of the sampled elements can be viewed as a Monte Carlo sample over the set of ordered point sequences (Metropolis & Ulam, 1949). Computationally, by applying a dense self-attention layer to each of the sub-sampled elements $\mathbf{X}^i$, the total complexity of evaluating l many self-attention layer is $O(l \cdot n_s^2)$. We however note that the l self-attention layers can be evaluated in parallel, which yields a trade-off between individual self-attention complexity $O(n_s^2)$ and computation time.

To gain intuition, consider the "limiting behaviours" of our random element sampling: taking $n_s = n + 1$ can be interpreted as taking the whole sequence with $l = 1$, *i.e.*, $\mathbf{X}^1 = \boldsymbol{X}$ which under dense attention would result in complexity $O(n^2)$. On the other end, if we take $n_s = 2$, we get $l = n$ pairs of points $|\mathbf{X}^i| = 2$; processing every such pair with dense self-attention results in n many $O(1)$ self-attention evaluations. Random element sampling with dense attention layers can be interpreted as an instance of sparse attention, see Fig. 1b.

4.2 Hamiltonian Self-Attention

The random element sampling discussed in the previous section reduces the computational complexity of dense self-attention-layers from $O(n^2)$ to $O(l \cdot n_s^2) = O(n^2/l)$ (as $(n_s - 1) \cdot l = n$) by processing each sampled set of points $\mathbf{X}^i$ through individual self-attention layers. Despite this improved computational complexity, the quadratic scaling of n can still be costly for point clouds.

As such, instead of evaluating each sampled element $\mathbf{X}^1, \ldots, \mathbf{X}^l$ with a dense self-attention layer, we propose a sparse attention layer. Sparse attention mechanisms can be formally defined via the attention patterns $\{\mathcal{A}_k\}_{k \in [n_s]}$, where $j \in \mathcal{A}_k$ implies that the j-th token will attend to the k-th token. We propose the use of an attention mechanism, dubbed as *Hamiltonian self-attention*, which is defined by the following attention patterns:

$$\mathcal{A}_k = \begin{cases} \{k, k+1\} & \text{if } 1 \leq k < n_s \\ \{k\} & \text{otherwise } k = n_s \end{cases}, \tag{4}$$

which ensures that the set of attention patterns $\{\mathcal{A}_k\}_{k \in [n_s]}$ define a *Hamiltonian path*. Indeed, if we fix a subset of elements $\mathbf{X}^i$, by starting at $\mathbf{X}_1^i$ and following the attended elements (ignoring self-attention $k \in \mathcal{A}_k$), we visit every token exactly once. Fig. 1c shows the corresponding attention matrix, where the Hamiltonian path corresponds to off-diagonal elements and self-attention corresponds to the diagonal elements, respectively.

For Hamiltonian self-attention, computing the attention mechanism according to Eq. 4 only requires $2n_s = O(n_s)$ many evaluations. Thus by using our proposed sparse attention for each $\mathbf{X}^1, \ldots, \mathbf{X}^l$, in comparison to dense attention, the computational complexity reduces from $O(n^2/l^2)$ to $O(n/l)$.

The proposed Hamiltonian self-attention mechanism is rather simple and general. For instance, in the general case sparsity patterns can be defined for each individual layer (resulting in an addition superscript for each A_k). Despite this, the attention patterns $\{A_k\}_{k \in [n_s]}$ satisfy important key assumptions for proving that the attention pattern will result in a sparse transformer that is a universal approximator (Yun et al., 2020, Assumption 1). In particular, by stacking $(n_s - 1)$ many attention layers, our Hamiltonian self-attention will allow any element to indirectly or directly attend all other element in a $\mathbf{X}^i$. The proposed Hamiltonian self-attention could also be viewed as a special case of window attention in Zaheer et al. (2020), where elements are linked undirectedly.

4.3 Sampled Sparse Attention Transformer

Given the setup of random element sampling and Hamiltonian self-attention, we can define our proposed sampled transformer for continuous set-to-set function approximation:

$$\text{SHead}_k^j(\mathbf{X}^i) = (\boldsymbol{W}_V^j \mathbf{X}^i_{\mathcal{A}_k}) \cdot \sigma_S[(\boldsymbol{W}_K^j \mathbf{X}^i_{\mathcal{A}_k})^T \boldsymbol{W}_Q^j \mathbf{X}^i_k] \tag{5a}$$

$$g^i(\mathbf{X}^i) = \mathbf{X}^i + \boldsymbol{W}_O \begin{bmatrix} \text{SHead}^1(\mathbf{X}^i) \\ \vdots \\ \text{SHead}^h(\mathbf{X}^i) \end{bmatrix} \tag{5b}$$

$$\text{SAttn}(\boldsymbol{X}) = g^l(\mathbf{X}^l) \circ g^{l-1}(\mathbf{X}^{l-1}) \circ \cdots \circ g^1(\mathbf{X}^1) \tag{5c}$$

$$\text{STB}(\boldsymbol{X}) = \text{SAttn}(\boldsymbol{X}) + \boldsymbol{W}_2 \cdot \text{ReLU}\left(\boldsymbol{W}_1 \text{SAttn}(\boldsymbol{X})\right). \tag{5d}$$

$$\tag{5e}$$

In Eq. 5c, composition is w.r.t. the induced linear maps from matrices given by Eq. 5b. The learnable parameters of the sampled transformer are the same as the usual dense transformer in Eq. 1.

As the attention pattern of each $\mathbf{X}^i$ forms a Hamiltonian path, and each $\mathbf{X}^i$ shares an element with the proceeding $\mathbf{X}^{\gamma(i)}$, the joint attention map makes a *Hamiltonian cycle* path. In other words, the shared index $\mathcal{R}_1^{\gamma(i+1)}$ in Eq. 3 links each individual Hamiltonian path given by Eq. 4, leading the attention matrix to form a *cycle attention* as shown in Fig. 1d. Furthermore, the permutation of elements in cycle attention corresponds to the swapping of nodes in the Hamiltonian cycle, with corresponding links and swapping of element values in the attention matrix, see in Fig. 1e. As a result, the combined randomization from using random element sampling and Hamiltonian self-attention can be thought of as sampling from the set of Hamiltonian cycle graphs from the complete attention graph, resulting in the *sampled attention* depicted in Fig. 1f.

Unlike dense attention, sparse attention patterns are not generally permutation invariant. Indeed, if we permute the columns of $\mathbf{X}^i$, the elements attended according to $\{\mathcal{A}\}_{k\in[n_s]}$ are not the same. As such, applying $\{\mathcal{A}_k\}_{k\in[n_s]}$ directly to $\boldsymbol{X}$ is not valid for point clouds, which requires a permutation invariant operation. However, in our case the sparse attention heads are being applied to *randomized* sub-sampled element sets $\mathbf{X}^i$. Ignoring computation, if we continue to sample the randomized elements $\mathbf{X}^i$ and average the resulting attention (w.r.t. the entire point set $\boldsymbol{X}$), the attention will converge to dense attention – through randomization of $\mathbf{X}^i$, the event that any non-self-edge appears in a sampled attention graph (as per Eq. 4) is equiprobable. This also holds when fixing the order of elements while applying randomly sampled Hamiltonian cycle attention. As such, the sampled transformer can be used to approximate a permutation invariant operator, and thus be used to approximate set-to-set functions.

Of course, sampling sufficiently many realizations of Hamiltonian cycle attention to converge to dense attention is impractical. Instead, in practice, we re-sample the attention pattern only for each batch and epoch. Although this may seem like a crude approximation to dense attention, similar methods are successful in Dropout (Srivastava et al., 2014), which even induces desirable model regularization. Furthermore, our empirical results indicate that sampled sparse attention closely approximates the more expensive (and infeasible at the typical point set scales) dense attention.

4.4 Sampled Transformer as a Universal Approximator

We formally guarantee the representation power of the proposed sampled transformer by proving universal approximation for set-to-set functions. As our sampled transformer Eq. 5c is similar to dense / sparse transformers presented by Yun et al. (2019; 2020), we follow their framework (Sec. 3.3) to prove our universal approximation property.

Corollary 1 (Sampled Transformer is a Universal Approximator)**.** *There exist sampled (sparse) Transformers that are universal approximators in the sense of Theorem 1.*

Table 1: Object classification on ModelNet40. Here [ST] denotes that model adopts the standard (dense) transformer, while [T] denotes all other transformers.

Supervised Methods	Accuarcy
PointNet (Qi et al., 2017a)	89.2%
PointNet++ (Qi et al., 2017b)	90.7%
PointCNN (Li et al., 2018)	92.5%
KPConv (Thomas et al., 2019)	92.9%
DGCNN (Wang et al., 2021)	92.9%
RS-CNN (Liu et al., 2019b)	92.9%
[T] PCT (Guo et al., 2021)	93.2%
[T] PVT (Zhang et al., 2021)	93.6%
[T] PointTransformer (Zhao et al., 2021)	93.7%
[T] Transformer (Yu et al., 2022)	91.4%

Self-Supervised Methods	Accuarcy
OcCo (Wang et al., 2021)	93.0%
STRL (Huang et al., 2021)	93.1%
IAE (Yan et al., 2022)	93.7%
[ST]Transformer-OcCo (Yu et al., 2022)	92.1%
[ST]Point-BERT (Yu et al., 2022)	93.2%
[ST]Point-MAE (Pang et al., 2022)	93.8%
[ST]**MAE-dense (ours)**	93.6%
[T]**MAE-sampled (ours)**	93.7%

To prove our Corollary, we extend the proof of Yun et al. (2019; 2020) by showing that our sparse attention mechanisms with random element sampling can also implement a selective shift operator. As a result, we show that the proposed sampled sparse attention transformer is a universal approximator in the context of set-to-set functions. See §E in the supplementary material for the full proof of the universal approximation property.

5 Experiments

We evaluate our proposed sampled attention in popular transformer-based frameworks as well as a basic setting. We compare our sampled attention (Fig. 1f) with dense attention via the pre-training and fine-tuning framework (Yu et al., 2022; Pang et al., 2022), where we pre-train our model on ShapeNet (Chang et al., 2015) via the reconstruction task, and further evaluate the performance on two downstream fine-tuning tasks: classification and few-shot learning in ModelNet40 (Wu et al., 2015). In addition, to eliminate the influence of other factors, we compared the dense, sparse, sampled, and kNN attention (Definition B.1) in a basic classification setting consisting of a transformer block with a single attention layer for feature aggregation, as well as a minimal number of MLPs for feature mapping. Finally, we compare the sampled attention with the kNN attention in the hierarchical grouping and merging structure following the Point-Transformer (Zhao et al., 2021).

5.1 Comparsion on Pre-training and Fine-tuning Framework

Pre-training. We adopted the masked auto-encoder (MAE) (He et al., 2022) to process the point cloud data, denoted as MAE-dense, for pre-training. This framework is the concurrent work with Point-MAE (Pang et al., 2022). Note that MAE-dense adopts dense-attention layers in its encoder and decoder network. To evaluate the effectiveness of our claimed contribution, we replace the dense-attention layer in MAE-dense with our sampled-attention layer (Fig. 1f) while keeping the other components fixed. It is denoted as MAE-sampled.

To pre-train the MAE-dense and MAE-sampled, we follow the standard train-test split of ShapeNet (Chang et al., 2015), which is also adopted by Pang et al. (2022); Yu et al. (2022), and develop the following training strategy. To begin with, each input point cloud consisting of 1024 points was divided into 64 groups / tokens of size 32 points each. The Furthest Points Sampling (FPS) and nearest neighbour search were adopted in tokenization (Yu et al., 2022). Tokens were further mapped to 256-dimensional latent vectors by MLP layers and max-pooling. In addition, we have 12 stacked transformers in the encoder (masking ratio of 70%) and 1 single transformer in the decoder, both with $h = 8$, $d = 32$ and $r = 256$. The batch size is 64 and the epoch number is 300. We used the AdamW (Loshchilov & Hutter, 2017) optimizer with cosine learning rate decay (Loshchilov & Hutter, 2016), an initial learning rate of 0.0005, and weight decay of 0.05.

Classification The pre-trained MAE-dense and MAE-sampled models are first evaluated on the classification task in ModelNet40 (Wu et al., 2015), with the standard training and testing splits defined in Yu et al. (2022); Pang et al. (2022). Specifically, we build the classifier by keeping the encoder structure and weights of the pre-trained MAE-dense and MAE-sampled models, followed by max-pooling as well as a fully connected layer of dimension $[256, 256, 40]$ to map the global

Table 2: Mean $\pm$ std. dev. accuracy (%) for 10 independent Few-shot classification experiments.

Methods	5-way, 10-shot	5-way,20-shot	10-way,10-shot	10-way, 20-shot
DGCNN-rand (Wang et al., 2021)	31.6 $\pm$ 2.8	40.8 $\pm$ 4.6	19.9 $\pm$ 2.1	16.9 $\pm$ 1.5
DGCNN-OcCo (Wang et al., 2021)	90.6 $\pm$ 2.8	92.5 $\pm$ 1.9	82.9 $\pm$ 1.3	86.5 $\pm$ 2.2
Transformer-rand (Yu et al., 2022)	87.8 $\pm$ 5.2	93.3 $\pm$ 4.3	84.6 $\pm$ 5.5	89.4 $\pm$ 6.3
Transformer-OcCo (Yu et al., 2022)	94.0 $\pm$ 3.6	95.9 $\pm$ 2.3	89.4 $\pm$ 5.1	92.4 $\pm$ 4.6
Point-BERT (Yu et al., 2022)	94.6 $\pm$ 3.1	96.3 $\pm$ 2.7	91.0 $\pm$ 5.4	92.7 $\pm$ 5.1
Point-MAE (Pang et al., 2022)	96.3 $\pm$ 2.5	97.8 $\pm$ 1.8	92.6 $\pm$ 4.1	**95.0** $\pm$ **3.0**
MAE-dense (ours)	95.9 $\pm$ 3.1	97.2 $\pm$ 2.1	90.8 $\pm$ 5.0	92.8 $\pm$ 3.9
MAE-sampled (ours)	**97.0** $\pm$ **2.3**	**98.3** $\pm$ **1.6**	**92.7** $\pm$ **5.4**	93.8 $\pm$ 3.5

Table 3: Object classification accuracy (%) for different attention mechanisms in the basic setting. OM denotes *out of memory*.

#Points	256	512	768	1024	2048	3072	4096	8192
MLP + FC (no attention)	85.96	86.24	85.43	85.96	85.84	86.61	86.32	86.13
Dense Attention	87.78	88.72	88.11	88.47	88.39	OM	OM	OM
Sparse Attention	87.09	88.03	87.54	87.74	87.58	87.42	87.42	87.58
Sampled Attention	87.34	87.93	87.66	88.03	87.82	87.18	87.46	87.73
kNN Attention	85.80	84.74	85.35	84.70	82.95	82.58	82.26	OM

token of a dimension of 256 to the 40 categories. Similar to Yu et al. (2022), we further data-augment the point cloud training set via random scaling and translation during training. As shown in Tab. 1, the proposed method achieved the second best performance compared with the most recent state-of-the-art alternatives. Our sampled attention can achieve an accuracy improvement of 0.1% when compared to dense attention, while reducing the complexity from $O(n^2)$ to $O(n)$.

Few Shot Learning The pre-trained MAE-dense and MAE-sampled models are also evaluated on a few shot learning task. Following Sharma & Kaul (2020); Wang et al. (2021); Yu et al. (2022); Pang et al. (2022), the few-shot learning adopted an k-way, m-shot training setting on the ModelNet40 (Wu et al., 2015) dataset, where k represents the number of randomly sampled classes and m the number of randomly sampled examples per class. The testing split is 20 randomly sampled unseen examples from each class. We set $k \in \{5, 10\}$ and $m \in \{10, 20\}$, and report the mean accuracy with standard deviation for 10 independent experiments. As shown in Tab. 2, our proposed MAE-sampled model outperformed all state-of-the-art methods on 3 out of 4 settings, while MAE-sampled consistently outperformed MAE-dense.

5.2 Comparsion on Basic Classification Setting

Our inputs are clouds of n points with 3D coordinates as position and its normal information as features. The feature and position are first transformed by two separate MLP layers with hidden dimensions $[64, 256]$, and then added together as the input of a single layer transformer with $h = 8$, $r = 256$, and $d = 32$, as per Eq. 1 and Eq. 5. The transformer output of $\mathbb{R}^{n\times 256}$ is then summarized by max-pooling to obtain a global feature with a dimension of 256, followed by a fully connected layer to map it to the category vector. Here we tested this basic pipeline with $n \in \{256, 512, 768, 1024, 2048, 3072, 4096, 8192\}$ for each of the dense, sparse, kNN, and the proposed sampled attention layers, including an additional case without attention layer (MLP+Full Connected layer) as the baseline.

As shown in Tab. 3 and Tab. 4, the model with dense attention layers achieves the best performance as it considers all $O(n^2)$ connections directly with relatively few parameters to train. However, it runs out of the 24 Gigabytes memory when the number of points $n \geq 3072$, due to the quadratic complexity. While both sparse and sampled transformers have a computational complexity of $O(n)$, our model with sampled attention outperformed the sparse one, in line with the strong theoretical guarantees we provide. We conjecture that the improvements of sampled transformer over the sparse transformer may indicate that the additional randomness (randomly shuffling points, w / o attention)

Table 4: Memory usage (Gb) for different attention mechanisms in the basic classification setting. All are trained on a single RTX 3090 with 24 Gb on board RAM. OM denotes *out of memory*.

#Points	256	512	768	1024	2048	3072	4096	8192
MLP + FC (no transformer)	0.9	0.9	0.9	1.0	1.1	1.1	1.2	1.5
Dense Attention	1.2	1.8	2.7	3.9	11.9	OM	OM	OM
Sparse Attention	1.0	1.1	1.2	1.3	1.7	2.1	2.5	4.3
Sampled Attention	1.0	1.1	1.2	1.3	1.7	2.1	2.5	4.2
kNN Attention	1.9	2.8	3.7	4.4	8.5	11.4	16.5	OM

Table 5: Classification accuracy (%) for sampled and kNN attention with hierarchical model structure.

#Layers	1	2	3	4	5
sampled attention	**74.55**	**88.0**	**90.5**	**91.0**	**91.8**
kNN attention	66.23	82.8	90.1	**91.0**	91.4

leads to a better approximation of the $O(n^2)$ connections in a manner analogous to Dropout (Srivastava et al., 2014). In addition, the transformer with kNN attention layers has the worst performance, as the permutation could not extend its receptive field. Finally, the memory usage comparison in Tab. 4 shows that the dense transformer has the largest memory usage due to its $O(n^2)$ complexity. The sparse transformer and sampled transformer have comparable memory usage due to the same $O(n)$ complexity.

5.3 Comparsion on Hierarchical Transformer Structure

We further compare our sampled attention with kNN attention by adopting the hierarchical structure for the classification task under the framework of Zhao et al. (2021). Each hierarchical layer is obtained by FPS, followed by the nearest neighbour search for the grouping, using MLPs with max-pooling for feature merging, and transformers for feature mapping. The grouping stage within each hierarchical layer summarizes the point cloud into key (subset) points.

The total hierarchical layer number is $t = 5$, the parameters for which we chose the number k of nearest neighbours $\{8, 16, 16, 16, 16\}$, strides $\{4, 4, 4, 4, 4\}$, self-attention feature dimensions $\{32, 64, 128, 256, 512\}$, and transformer blocks $\{2, 3, 4, 6, 3\}$. The scalar attention (Eq. 1 or Eq. 5) is adopted specifically for comparison. Results shown in Tab. 5 demonstrate that our sampled attention outperforms the kNN attention in line with our randomly sampled receptive field. Furthermore, the performance of the kNN layer improved greatly from $t = 1$ to $t = 2$ and from $t = 2$ to $t = 3$ as its receptive field extends due to the multiple hierarchical layers. Finally, kNN with vector attention (Yu et al., 2022) (reported in Tab. 1 on the PointTransformer row) achieved a better performance, in line with the observation that replacing the softmax with learnable MLPs γ in the transformer can easier make kNN attention a universal approximator of continuous functions. Detailed analysis is provided in §B.1 in the supplementary material. The performance difference between scalar attention and vector attention is shown in the Tab. 7 of Yu et al. (2022), and is also analyzed in Yun et al. (2020).

6 Conclusion

In this paper, we present an $O(n)$ complexity sparse transformer – *sampled transformer* – which directly handles point set data. By relating the permutation of set elements to the sampling of Hamiltonian cycle attention, we relieve the model of inappropriate permutation variance. The result is a sampled attention scheme that implements Monte Carlo simulation to approximate a dense attention layer with a prohibitive $O(n^2)$ number of connections. To guarantee the representation power of the proposed sampled transformer, we showed that it is a universal approximator of set-to-set functions. Motivated also by the strong empirical performance that our model achieves, we hope this work will help to shed light on the sparse transformer in dealing with set data.

REFERENCES

Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

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

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.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

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, 2021.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. *arXiv preprint arXiv:1902.09113*, 2019.

Xian-Feng Han, Yi-Fei Jin, Hui-Xian Cheng, and Guo-Qiang Xiao. Dual transformer for point cloud analysis. *IEEE Transactions on Multimedia*, 2022.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6535–6545, 2021.

Anastasis Kratsios, Behnoosh Zamanlooy, Tianlin Liu, and Ivan Dokmanić. Universal approximation under constraints is possible with transformers. *arXiv preprint arXiv:2110.03303*, 2021.

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*, pp. 8500–8509, 2022.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

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.

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*, volume 33, pp. 8778–8785, 2019a.

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*, pp. 8895–8904, 2019b.

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*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3164–3173, 2021.

Kirill Mazur and Victor Lempitsky. Cloud transformers: A universal approach to point cloud processing tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10715–10724, 2021.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2906–2917, 2021.

Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. *arXiv preprint arXiv:2203.06604*, 2022.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

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, 2017b.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.

Michael E Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, pp. 3515–3530. PMLR, 2022.

Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. *Advances in Neural Information Processing Systems*, 33:7212–7221, 2020.

Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James Tin-Yau Kwok. Sparsebert: Rethinking the importance analysis in self-attention. In *International Conference on Machine Learning*, pp. 9547–9557. PMLR, 2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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*, pp. 6411–6420, 2019.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

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*, pp. 1588–1597, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9782–9792, 2021.

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*, pp. 1912–1920, 2015.

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*, pp. 4606–4615, 2018.

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)*, pp. 87–102, 2018.

Siming Yan, Zhenpei Yang, Haoxiang Li, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. Implicit autoencoder for point cloud self-supervised representation learning. *arXiv preprint arXiv:2201.00785*, 2022.

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*, pp. 3323–3332, 2019a.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019b.

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*, pp. 19313–19322, 2022.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for point cloud learning. *arXiv preprint arXiv:2108.06076*, 2021.

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*, pp. 16259–16268, 2021.

A Notations

f	a continuous function
g	transformer
$\overline{g}$	modified transformer
$\mathcal{F}$	the class of continuous sequence-to-sequence function
$\mathcal{F}_S$	the class of continuous set-to-set function
$\overline{\mathcal{F}}$	the class of piece-wise constant sequence-to-sequence function
$\overline{\mathcal{F}}_S$	the class of piece-wise constant set-to-set function
$\mathcal{T}^{h,m,r}$	the class of (sparse) transformers with h attention heads, m head size, and hidden layer width r
$\overline{\mathcal{T}}^{h,m,r}$	the class of the modified transformers with h attention heads, m head size, and hidden layer width r
σ_S	softmax activation
σ_H	hardmax activation
ℓ_p	p norm
$\mathbb{G}_\delta$	grid $\{0, \delta, \dots, 1-\delta\}^{d\times n}$
$\mathbb{G}_\delta^+$	extend grid $\{-\delta^{-nd}, 0, \delta, \dots, 1-\delta\}^{d\times n}$
n	number of points/elements/tokens
d	point/element/token feature size
m	head size
h	heads number
r	hidden layer width
δ	step size
$\boldsymbol{X}$	transformer input
$\mathbf{X}^i$	i-th subset of transformer input
$\boldsymbol{P}$	xyz coordinates for point cloud (set)
$\boldsymbol{E}$	positional embedding
$\boldsymbol{L}$	quantized transformer input
$\boldsymbol{A_L}$	desired output for the input $\boldsymbol{L}$
$\boldsymbol{W}_V^i$	value parameter in i-th single-head attention layer
$\boldsymbol{W}_K^i$	key parameter in i-th single-head attention layer
$\boldsymbol{W}_Q^i$	query parameter in i-th single-head attention layer
$\boldsymbol{W}_O$	multi-head attention parameter
$\boldsymbol{W}_1$	feed-forward layer parameter
$\boldsymbol{W}_2$	feed-forward layer parameter
$\boldsymbol{W}_p$	parameter for position embedding

$\boldsymbol{u}$	query, key, and value parameter used in universal approximation proof
$\boldsymbol{e}^{(1)}$	indicator vector $(1, 0, 0, \dots, 0) \in \mathbb{R}^d$
$\mathbf{1}_n$	vector with all ones $(1, \dots, 1) \in \mathbb{R}^n$
$\mathbf{0}_n$	vector with all zeros $(0, \dots, 0) \in \mathbb{R}^n$
$\mathrm{Head}^i(\cdot)$	i-th single-head attention layer
$\mathrm{SHead}^i(\cdot)$	i-th sparse/sampled single-head attention layer
$\mathrm{Attn}(\cdot)$	multi-head attention layer
$\mathrm{SAttn}(\cdot)$	multi-head attention layer with sampled sparse attention
$\mathrm{TB}(\cdot)$	transformer block
$\mathrm{STB}(\cdot)$	sampled transformer block
$t(\cdot)$	a series of any number of transformer blocks
$q_c(\cdot)$	contextual mapping
$\boldsymbol{\Psi}(\cdot; b_Q, b'_Q)$	selective shift operation
$\psi(\cdot; b_Q)$	a single-head attention in selective shift operation
$d_c(\cdot, \cdot)$	distance between two functions

B Additional Information on the Basic Classification Setting

B.1 kNN Transformer

Definition B.1 (kNN Attention)**.** *For $k \in [n]$, kNN attention has the attention pattern $\mathcal{A}_k$ = kNN(k) for all points, where kNN($\cdot$) represents the Euclidean k-nearest neighbourhood of the input.*

Definition B.2 (kNN Transformer)**.** *The kNN transformer is the transformer defined as in Eq. 1, but with the kNN attention of definition B.1.*

In addition, in the case of vector attention (Eq. 3 in (Zhao et al., 2021)), universal approximation holds as the learnable mapping $\gamma(\cdot)$ (an MLP) is a universal approximator. This may helps to explain why vector attention could outperform scalar attention in Tab. 7 of (Zhao et al., 2021).

Finaly, in Tab. 3, the performance of the kNN transformer drops with the increasing number of points. This is because as the point number increase, the fix k nearest neighbor number is relatively reduced. As a result, the receptive field shrink. So the performance drops.

B.2 In comparison with Inducting Points (Set Transformer)

We additionally compared the proposed sampled attention with learnable inducting points strategy (Lee et al., 2019) in Tab.6. The inducting points here are implemented by simply replacing the multi-heads self-attention transformer block in Eq. 5e with the Induced Set Attention Block (ISAB) in Eq. (9) of Lee et al. (2019). And the positional embedding is added in the key and value input as per our sampled attention. Our implementation of the basic classification in Sec. 5.2 is different from the one in (Lee et al., 2019) with respect to the data pre-processing: our data pre-processing is in line with Zhao et al. (2021); Yu et al. (2022), while Lee et al. (2019) follow Zaheer et al. (2017) without positional embedding. As we can see in Tab. 6, our proposed sampled attention outperformance the inducting point strategy (Lee et al., 2019) with linear complexity in the attention matrix.

As the performance of Lee et al. (2019) on the two implementations is quite different, we further compared the sampled attention and inducting points strategy in the implementation provided by the official implementation of Lee et al. (2019). To begin with, our proposed sampled attention could be applied to the inducting points strategy directly to reduce its complexity from $O(mn)$ to $O(n)$,

Table 6: Additional object classification accuracy (%) for different attention mechanisms in the basic setting. OM denotes *out of memory*.

#Points	256	512	768	1024	2048	3072	4096	8192
MLP + FC (no attention)	85.96	86.24	85.43	85.96	85.84	86.61	86.32	86.13
Inducting Points (Lee et al., 2019)	84.21	81.25	82.55	81.57	80.96	76.18	75.13	75.65
Stratified Strategy (Lai et al., 2022)	87.21	87.62	86.69	85.99	85.34	84.32	OM	OM
Sampled Attention	**87.34**	**87.93**	**87.66**	**88.03**	**87.82**	**87.18**	**87.46**	**87.73**

Table 7: Object classification in the setting of Lee et al. (2019) measured accuracy (%).

#Points	100	200	1000	2000	3000	5000
ISAB(16) + PMA	80.52	85.38	84.43	85.99	85.49	86.99
ISAB(16) + PMA + sampled attention (ours)	81.25	82.65	84.15	85.04	84.48	86.49

where n is the number of input points and m is a learnable inducting points number. Specifically, we use the sampled attention to replace the dense attention in the Induced Set Attention Block(ISAB) from Eq. 9 of Lee et al. (2019). However, as the inducting points and points have different physical meanings, also as the inducting points number m (query in the self-attention) is not equal to the input points number n (key and value), our Hamiltonian cycle attention could not be applied directly. We instead applied a different version of sampled attention by randomly sampling two elements per row in the dense attention matrix. This is a loose version of sampled attention as no Hamiltonian cycle is constructed. The results could be found in Tab. 7. As we can see, our proposed sampled attention is still comparable with the set transformer but with less computational complexity.

B.3 In comparison with Stratified Strategy

The window-based transformer is another important branch of exploring the representation power of the transformer. Combined with the hierarchical backbone, it has been widely used in processing 2D images, languages, and 3D point clouds, such as Liu et al. (2021); Lai et al. (2022). The window-based transformer is proposed to learn the cross-window relationships as well as the non-overlapping local relationship.

Here we compared our proposed sampled attention with the Stratified strategy from Figure 3 of Lai et al. (2022) in Tab. 6. The Stratified strategy could be viewed as a combination of dense and sparse keys obtained by the window partition of different sizes. It is an efficient design for learning token relationships in the hierarchical backbone. However, in the single-layer setting, directly learning $O(n^2)$ connections in the attention matrix may be a better solution as it could reach the full receptive field. As our proposed sampled attention mechanism could estimate $O(n^2)$ connections by implementing the Monto Carlo simulation, we outperformed the Stratified strategy in the basic classification setting as per Tab. 6.

C Amortized Clustering with Mixture of Gaussians

We additionally tested the proposed sampled attention in 2D set datasets in the encoding-decoding framework introduced by (Lee et al., 2019). And the task is about using a neural network to learn the parameters of the mixture Gaussian distribution from the input set data.

To begin with, the mixture Gaussian distribution is defined by a weighted sum of k number of Gaussian distribution. Given a dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$, the log-likelihood of the mixture Gaussian distribution is defined as follows:

$$\log p(\boldsymbol{X}; \boldsymbol{\theta}) = \sum_{i=1}^{n} \log \sum_{j=1}^{k} \pi_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_i; \mathrm{diag}(\boldsymbol{\sigma}_j^2)). \tag{6}$$

Generally, the parameters of the mixture Gaussian distribution are inferred by maximizing the log-likelihood $\theta^*(\boldsymbol{X}) = \arg\max_\theta \log p(\boldsymbol{X}; \theta)$ using Expectation-Maximisation (EM) algorithm as the

Table 8: Amortized clustering results. The number in ISAB($\cdot$) indicates the number of learnable inducting points used in ISAB as per Lee et al. (2019). The evaluation metric LLO/data is the average log-likelihood value, and LL1/data is the average log-likelihood value after a single EM update (implemented by scikit-learn package (Buitinck et al., 2013)).

Architecture	LL0/data	LL1/data
rFF + Pooling	-2.0006 $\pm$ 0.0123	-1.6186 $\pm$ 0.0042
ISAB(16) + PMA	-1.5034 $\pm$ 0.0072	-1.4908 $\pm$ 0.0044
ISAB(16) + PMA + sampled attention (ours)	-1.5663 $\pm$ 0.0074	-1.5272 $\pm$ 0.0052

closed-form solution could not be inferred directly by setting the gradient equals to zero. Here we instead use the transformer to infer $\theta^*(\boldsymbol{X})$. Specifically, given the input, the neural network f outputs mixture Gaussian parameters $f(\boldsymbol{X}) = \{\pi(\boldsymbol{X}), \{\mu_j(\boldsymbol{x}), \sigma_j(\boldsymbol{X})\}_{j=1}^k\}$ by maximing the log likelihood in Eq. 6 (and replacing all parameters as functions of $\boldsymbol{X}$).

The 2D set data $\boldsymbol{X}$ is randomly sampled from a given mixture Gaussian distribution with $k = 4$. And the number of elements n is randomly sampled from $[100, 500]$. Namely, when setting the dimension of Gaussian distribution as 2, each sampled point could be viewed as a 2D data point, so the sampled collection is a 2D set dataset.

The baseline we compared with is the Set transformer (Lee et al., 2019) with two Induced Set Attention Block(ISAB) in the encoder, one Multi-head Attention (PMA) and two Set Attention Block (SAB) in the decoder, as per the official implementation. The inducting points refer to the additional learnable points $\boldsymbol{I} \in \mathbb{R}^{m \times d}$ proposed in Eq. 9 of (Lee et al., 2019), with d dimension and m number of inducting points. Here we have a mixture usage of points, tokens, and elements to represent a single sampled data point x_i.

As the computation complexity of the inducting points block (ISAB) is $O(nm)$, our sampled attention may be adopted in the ISAB to reduce the computation complexity to $O(n)$. However, as the number of inducting points m (regarded as the query in Lee et al. (2019)) is not equal to the number of input points n (regarded as key and value) (in fact inducting points and points have different physical meanings), our Hamiltonian cycle attention could not be applied directly. In fact, the dense attention matrix in the inducting points layer is $m \times n$ rather than $n \times n$. We instead applied a different version of sampled attention by randomly sampling two elements per row in the attention matrix. This is a loose version of sampled attention as no Hamiltonian cycle is constructed. As we can see in Tab. 8, the sampled attention could be a plug-in module to replace the dense attention in the inducting points structure with competitive performance but theoretically less computational complexity.

D Transfer Learning

We additionally included the transfer learning as a fine-tuning classification task with respect to the pre-training and fine-tuning framework in Sec. 5.1, to demonstrate that the proposed sampled transformer also has a good transfer ability. The fine-tuning task is implemented on the ScanObjectNN (Uy et al., 2019) dataset with 2902 point clouds from 15 categories. We follow the data pre-processing and fine-tuning setting from Point-BERT (Yu et al., 2022) with the same three variants: OBJ-BG, OBJ-ONLY, and PB-T50-RS. As we can see in Tab. 9, our sampled attention layer achieved a competitive performance in comparison with dense attention while reaching state-of-the-art performance.

E Universal Approximator Proof

A proof of Corollary 1 follows the steps described in § 3.3. As we only changed the dense/sparse attention to the sampled attention, the steps ① and ③ in § 3.3 remain the same as Yun et al. (2019; 2020) and found in the §C and F in Yun et al. (2020). Here we need only cover the proof of step ②.

First, we have $\mathcal{F}_S(\cdot)$ is the class of continuous set-to-set function, and $\overline{\mathcal{F}}_S(\cdot)$ is the class of piece-wise constant set-to-set function.

Table 9: Transfer learning on the classification task, measured by the Accuracy (%).

Methods	OBJ-BG	OBJ-ONLY	PB-T50-RS
PointNet (Qi et al., 2017a)	73.3	79.2	68.0
SpiderCNN (Xu et al., 2018)	77.1	79.5	73.7
PointNet++ (Qi et al., 2017b)	82.3	84.3	77.9
PointCNN (Li et al., 2018)	86.1	85.5	78.5
DGCNN(Wang et al., 2021)	82.8	86.2	78.1
BGA-DGCNN (Uy et al., 2019)	-	-	79.7
BGA-PN++ (Uy et al., 2019)	-	-	80.2
Point-BERT (Yu et al., 2022)	87.43	88.12	83.07
Point-MAE (Pang et al., 2022)	90.02	88.29	**85.18**
MAE-dense (ours)	**90.36**	88.50	83.41
MAE-sampled (ours)	89.68	**88.81**	82.44

Lemma 2 (Modified Universal Approximation.). *For each* $\overline{f} \in \overline{\mathcal{F}}_S(\delta)$ *and* $1 \leq q < \infty$, $\exists \overline{g} \in \overline{\mathcal{T}}^{2,1,1}$ *such that* $\overline{f}(\boldsymbol{X}) = \overline{g}(\boldsymbol{X})$ *for all* $\boldsymbol{X} \in \mathbb{D}$.

Without loss of generality, here $\mathbb{D} \in [0,1)^{d\times n}$. As in (Yun et al., 2019; 2020) The proof of Lemma 2 could then be separated into four steps:

1. Use the positional embedding $\boldsymbol{E}$ in § 3.2 such that each column of the input $\boldsymbol{X}_k + \boldsymbol{E}_k$ are in disjoint intervals.
2. The input $\boldsymbol{X}+\boldsymbol{E}$ is quantized into $\boldsymbol{L}$ with values in $\{0, \delta, \dots, n-\delta\}$ by a series of modified feed-forward layers.
3. The *contextual mapping q* defined in Definition 3.1 is implemented by a series of modified sampled multi-head self-attention layers (modified version of Eq. 5c) with the input of $\boldsymbol{L}$.
4. Another series of modified feed-forward layers implements the *value mapping* such that each element in the unique id $q(\boldsymbol{L})$ is mapped to the desired output $\boldsymbol{A}_{\boldsymbol{X}}$.

As modified feed-forward layers are all the same as in Yun et al. (2020), the definition and proof of step 2 is available in §D.2 and E.1 in (Yun et al., 2020), while the definition and proof of step 4 could be found in the §D.4 and E.3 in (Yun et al., 2020). Here we mainly explain steps 1 and 3.

E.1 Positional Embedding

The positional input for point sets in its xyz coordinate $\boldsymbol{P} \in \mathbb{R}^{3\times n}$. We adopted a matrix $\boldsymbol{W}_p \in \mathbb{R}^{d\times 3}$ (a permutation invariant operation) such that the input of the sampled transformer will be $\boldsymbol{X} + \boldsymbol{E} = \boldsymbol{X} + \boldsymbol{W}_p\boldsymbol{P}$. And there exists a case such that:

$$\boldsymbol{E}_1 = (n-1)\mathbf{1}_n, \text{ and } \boldsymbol{E} = (i-2)\mathbf{1}_n, \text{ for } i \in [2:n]. \tag{7}$$

In this case, the first column will be $(\boldsymbol{X}+\boldsymbol{E})_1 \in [n-1, n)^d$, and $(\boldsymbol{X}+\boldsymbol{E})_i \in [i-2, i-1)^d$ for $i \in [2:n]$. So the requirement of step 1 is satisfied, that each column lies in disjoint intervals.

E.2 Contextual Mapping for Stacked Multi-Heads self-Attention Layers

After the step 2, the quantized input $\boldsymbol{L}$ will be in the set $\mathbb{H}_\delta \subset \mathbb{R}^{d\times n}$, such that:

$$\mathbb{H}_\delta := \{\boldsymbol{G} + \boldsymbol{E} \in \mathbb{R}^{d\times n} | \boldsymbol{G} \in \mathbb{G}_\delta\}, \tag{8}$$

with $\mathbb{G}_\delta := \{0, \delta, \dots, 1-\delta\}$. Then the adaptive selective shift operation Ψ is defined so that the learnable parameter $\boldsymbol{u}^T \in \mathbb{R}^d$ could map $\boldsymbol{u}^T\Psi(\boldsymbol{L})$ into unique scalars (ids). Finally, with the help of the all-max-shift operation Ω, the output of a series of those two operations will be a scalar in disjoint intervals w.r.t each column of $\boldsymbol{L}$, as well as different inputs $\boldsymbol{L}$ and $\boldsymbol{L}'$, thereby implementing the contextual mapping in Definition. 3.1.

Adaptive Selective Shift Operation. With a 2 heads and 1 hidden layer width modified multi-heads attention layer, the adaptive selective shift operation $\Psi(\cdot)$ may be defined as:

$$\Psi^l(\boldsymbol{L}^l; c, b_Q, b'_Q) := \boldsymbol{L}^l + c[\mathbf{1}^1_{n_s} - \mathbf{1}^1_{n_s}] \begin{bmatrix} \psi^l(\boldsymbol{L}^l; b_Q) \\ \psi^l(\boldsymbol{L}^l; b'_Q) \end{bmatrix} \tag{9a}$$

$$\begin{aligned} \psi^l(\boldsymbol{L}^l; b_Q)_k &= \boldsymbol{u}^T \boldsymbol{L}_{\mathcal{A}^l_k} \sigma_H \left[(\boldsymbol{u}^T \boldsymbol{L}_{\mathcal{A}^l_k})^T (\boldsymbol{u}^T \boldsymbol{L}^l_k - b_Q) \right] \\ &= \begin{cases} \max_{j \in \mathcal{A}^l_k} \boldsymbol{u}^T \boldsymbol{L}^l_j & \text{if } \boldsymbol{u}^T \boldsymbol{L}^l_k > b_Q \\ \min_{j \in \mathcal{A}^l_k} \boldsymbol{u}^T \boldsymbol{L}^l_j & \text{if } \boldsymbol{u}^T \boldsymbol{L}^l_k < b_Q, \end{cases} \end{aligned} \tag{9b}$$

where we assign query, key, and value parameters as $\boldsymbol{u}^T$, and we introduced the superscript l to denote different attention layers of self-attention layer l. With the help of hardmax, the k-th row of the attention matrix will be one-hot vectors to select the max or min vector in $\mathcal{A}^l_k$. $\boldsymbol{W}_O = c[\mathbf{1}^1_{n_s} - \mathbf{1}^1_{n_s}] \in \mathbb{R}^{n_s \times 2}$ is used to make sure only the first element in feature dimension are changed in selective shift operation. Specifically, the $1, k$-entity of the self-attention output reads:

$$\Psi^l(\boldsymbol{L}^l; c, b_Q, b'_Q)_{1,k} = \overline{L}^l_{1,k} + c \left(\psi^l(\boldsymbol{L}^l; b_Q)_k - \psi^l(\boldsymbol{L}^l; b'_Q)_k \right) \tag{10}$$

$$= \begin{cases} \overline{L}^l_{1,k} + c \left(\max_{j \in \mathcal{A}^l_k} \boldsymbol{u}^T \boldsymbol{L}^l_j - \min_{j \in \mathcal{A}^l_k} \boldsymbol{u}^T \boldsymbol{L}^l_j \right) & \text{if } b_Q < \boldsymbol{u}^T \boldsymbol{L}^l_k < b'_Q, \\ \overline{L}^l_{1,k} & \text{if } \boldsymbol{u}^T \boldsymbol{L}^l_k \notin [b_Q, b'_Q]. \end{cases} \tag{11}$$

Without loss of generality, the sampled transformer in §. 4.3 may be viewed as a series of stacked masked attention $\mathcal{A}^i$ for $i \in [n]$, such that:

$$\mathcal{A}^i_{k=1+(i-2+n \mod n)} = \{i, 1 + (i - 2 + n \mod n)\} \tag{12a}$$

$$\mathcal{A}^i_{k=i} = \{i\} \tag{12b}$$

$$\mathcal{A}^i_{k \not\subset \{i, i-1 \mod n\}} = \{\}, \tag{12c}$$

for $k \in [n]$. This is in fact the n point pairs in the Hamiltonian cycle. So the stack of all the masked attention is the cycle attention in Fig. 1d reflected across the diagonal line. Then the Eq. 5d will be

$$\text{SAttn}(\boldsymbol{L}) = g^n(\boldsymbol{L}_{\mathcal{A}_n}) \circ g^{n-1}(\boldsymbol{L}_{\mathcal{A}_{n-1}}) \circ \cdots \circ g^1(\boldsymbol{L}_{\mathcal{A}_1}), \tag{13}$$

noting that the updated column for previous g^i will be applied to the next g^{i+1}. In conclusion, the contextual mapping holds as the masked attention $\mathcal{A}^i$ is designed to aggregate information from all n elements / tokens by applying the $g(\cdot)$ about $O(n)$ times, which matches the design of (Yun et al., 2020).

Now consider $\boldsymbol{u}^T = (1, \delta^{-1}, \delta^{-2}, \ldots, \delta^{-d+1})$, the mapping $l_i = p_s(\boldsymbol{L}_i) = \boldsymbol{u}^T \boldsymbol{L}_i$ is bijective as all input point features $\boldsymbol{L}_i$ are different with at least one element having a gap of δ. In addition, without loss of generality, the order $l_2 < l_3 < \ldots < l_n < l_1$ holds as in (Yun et al., 2020) because of the positional embedding $\boldsymbol{E}$. Further, as each l_i has δ^{-d} intervals, and as the n tokens are disjoint with each other, we need $n\delta^{-}d$ adaptive selective operations to achieve the bijective mapping of unique ids.

First δ^{-d} selective shift operations. The first δ^{-d} layers are all applied to the second column (token) within $l_2 \in \left[0 : \delta : \delta^{-d+1} - \delta\right]$, and each selective shift operation will match one interval within $b_Q = b - \frac{\delta}{2}, b'_Q = b + \frac{\delta}{2}$ for $b \in \left[0 : \delta : \delta^{-d+1} - \delta\right]$. Also $\mathcal{A}^2$ is in fact $\mathcal{A}^2_1 = \{1\}$, $\mathcal{A}^2_2 = \{1, 2\}$, and is empty otherwise. So all δ^{-d} layers are only applied on the first two token embeddings, then the maximum value is l_1 and the minimum value is l_2. We have the output after those selective shift operations:

$$\tilde{l}_2 = l_2 + \delta^{-d}(\max_{j \in \mathcal{A}^1_2} l_j - \min_{j \in \mathcal{A}^1_2} l_j) = l_2 + \delta^{-d}(l_1 - l_2), \tag{14}$$

where with constant value $c = \delta^{-d}$ in Eq. 9b. Note that $\tilde{l}_2 > l_1$ because

$$l_2 + \delta^{-d}(l_1 - l_2) > l_1 \Leftrightarrow (\delta^{-d} - 1)(l_1 - l_2) > 0, \tag{15}$$

which is true. So the current order becomes $l_3 < l_4 < \ldots < l_n < l_1 < \tilde{l}_2$. So in the next δ^{-d} selective shift operations, the maximum value will be $\tilde{l}_2$ and the minimum will be l_3.

Second δ^{-d} selective shift operations. The next δ^{-d} layers will be applied on the third column (token embedding) within intervals $l_3 \in \left[\sum_{i=0}^{d-1} \delta^{-i} : \delta : \sum_{i=0}^{d-1} \delta^{-i} + \delta^{-d+1} - \delta\right]$ which results in

$$\tilde{l}_3 = l_3 + \delta^{-d}(\tilde{l}_2 - l_3) = l_3 + \delta^{-d}(l_2 - l_3) + \delta^{-2d}(l_1 - l_2), \tag{16}$$

which is again $\tilde{l}_3 > \tilde{l}_2$ because

$$l_3 + \delta^{-d}(\tilde{l}_2 - l_3) > \tilde{l}_2 \Leftrightarrow (\delta^{-d} - 1)(\tilde{l}_2 - l_3) > 0. \tag{17}$$

So we have a new maximum $\tilde{l}_3$ and new minimum l_4.

Repeat after $(n-1)\delta^{-d}$ operations. The next δ^{-d} will operate on the fourth column. After all $(n-1)\delta^{-d}$ operations we have

$$(n-1)\sum_{i=0}^{d-1} \delta^{-i} \le l_1 < \tilde{l}_2 < \ldots < \tilde{l}_n. \tag{18}$$

For j-th column, we will have the output

$$\tilde{l}_1 = l_1, \tag{19a}$$
$$\tilde{l}_2 = l_2 + \delta^{-d}(l_1 - l_2), \tag{19b}$$
$$\tilde{l}_j = l_j + \sum_{k=1}^{j-2} \delta^{-kd}(l_{j-k} - l_{j-k+1}) + \delta^{-(j-1)d}(l_1 - l_2). \tag{19c}$$

And we also know the interval of each l_i

$$l_1 \in [(n-1)\Delta : \delta : (n-1)\Delta + \delta^{-d+1} - \delta] \tag{20}$$
$$l_i \in [(i-2)\Delta : \delta : (i-2)\Delta + \delta^{-d+1} - \delta], \tag{21}$$

with $\delta^{-d+1} - \delta < \Delta := \sum_{i=0}^{d-1} \delta^{-i} = \frac{\delta^{-d}-1}{\delta^{-1}-1} \le \delta^{-d} - 1 \Rightarrow 0 < \delta \le \frac{1}{2}$. So we have

$$l_1 - l_2 \in [(n-1)\Delta - \delta^{-d+1} + \delta : \delta : (n-1)\Delta + \delta^{-d+1} - \delta] \tag{22}$$
$$l_i - l_{i+1} \in [-\Delta - \delta^{-d+1} + \delta : \delta : -\Delta + \delta^{-d+1} - \delta] \text{ for } i \in \{2, 3, \ldots, n-1\}. \tag{23}$$

Then the interval of outputs are

$$\tilde{l}_1 \in [(n-1)\Delta, (n-1)\Delta + \delta^{-d+1} - \delta] \tag{24}$$
$$\tilde{l}_2 \in [(n-1)\Delta\delta^{-d} - \delta^{-2d+1} + \delta^{-d+1}, (n-1)\Delta\delta^{-d} + \delta^{-2d+1} - \delta] \tag{25}$$
$$\begin{aligned}\tilde{l}_i \in [&(i-2)\Delta - \sum_{k=1}^{i-2} \delta^{-kd}\Delta - \sum_{k=1}^{i-2} \delta^{-kd}(\delta^{-d+1} - \delta) + \delta^{-(i-1)d}(n-1)\Delta - \delta^{-(i-1)d}(\delta^{-d+1} - \delta),\\ &(i-2)\Delta + \delta^{-d+1} - \delta - \sum_{k=1}^{i-2} \delta^{-kd}\Delta\\ &\quad + \sum_{k=1}^{i-2} \delta^{-kd}(\delta^{-d+1} - \delta) + \delta^{-(i-1)d}(n-1)\Delta + \delta^{-(i-1)d}(\delta^{-d+1} - \delta)],\end{aligned} \tag{26}$$

and to check whether intervals are disjoint or not, we take the difference between the lower bound of $\tilde{l}_{i+1}$ and the upper bound of $\tilde{l}_i$

$$\tilde{l}^l_{i+1} - \tilde{l}^u_i = \Delta - \delta^{-(i-1)d}\Delta + (\delta^{-id} - \delta^{-(i-1)d})(n-1)\Delta - (\delta^{-d+1} - \delta) \tag{27}$$

$$- \delta^{-(i-1)d}(\delta^{-d+1} - \delta) - 2\sum_{k=1}^{i-2} \delta^{-kd}(\delta^{-d+1} - \delta) \tag{28}$$

$$- \delta^{-id}(\delta^{-d+1} - \delta) - \delta^{-(i-1)d}(\delta^{-d+1} - \delta) \tag{29}$$

$$= \left[1 - n\delta^{-(i-1)d} + (n-1)\delta^{-id}\right]\Delta$$

$$- \left(\frac{1+\delta^{-d}}{1-\delta^{-d}} - \frac{2\delta^{-d}}{1-\delta^{-d}}\delta^{-(i-2)d} + 2\delta^{-(i-1)d} + \delta^{-id}\right)(\delta^{-d+1} - \delta) \tag{30}$$

$$\geq \left[\frac{2\delta^{-d}}{\delta^{-d}-1} - \frac{2\delta^{-d}}{\delta^{-d}-1}\delta^{-(i-2)d} - (n+2)\delta^{-(i-1)d} + (n-2)\delta^{-id}\right](\delta^{-d+1} - \delta) \tag{31}$$

$$\geq \delta^{-(i-2)d}\left[-\frac{2\delta^{-d}}{\delta^{-d}-1} - (n+2)\delta^{-d} + (n-2)\delta^{-2d}\right](\delta^{-d+1} - \delta) \tag{32}$$

$$\geq \delta^{-(i-2)d}\left[-4 - (n+2)\delta^{-d} + (n-2)\delta^{-2d}\right](\delta^{-d+1} - \delta), \tag{33}$$

which is not guaranteed to be above 0, so the addition operations should be introduced.

Further, the adaptive shift operation is a one-to-one map as the map $\boldsymbol{L}_k \mapsto \boldsymbol{u}^T\boldsymbol{L}_k$ is one-to-one, and the permutation of columns is one-to-one, and so it sufficies to prove that the map $[l_1 \cdots l_n] \mapsto \tilde{l}_k$ is also one-to-one. See the detailed analysis in §E.2.3 in (Yun et al., 2020).

Preliminaries. As in (Yun et al., 2020), the upper bound for the unique id $\tilde{l}_i$ is:

$$\tilde{l}_i := l_i + \sum_{j=1}^{i-2} \delta^{-jd}(l_{i-j} - l_{i+1-j}) + \delta^{-(i-1)d}(l_1 - l_2)$$

$$\leq l_i + \delta^{-d}\sum_{j=1}^{i-2}(l_{i-j} - l_{i+1-j}) + \delta^{-(i-1)d}(l_1 - l_2)$$

$$= l_i + \delta^{-d}(l_2 - l_i) + \delta^{-(i-1)d}(l_1 - l_2)$$

$$= \delta^{-(i-1)d}l_1 - (\delta^{-(i-1)d} - \delta^{-d})l_2 - (\delta^{-d} - 1)l_i \tag{34}$$

$$\leq \delta^{-(i-1)d}l_1 \leq \delta^{-(i-1)d}\left((n-1)\Delta + \delta^{-d+1} - \delta\right) \tag{35}$$

$$\leq \delta^{-(i-1)d}(i-1+\delta)(\delta^{-d} - 1) \leq n\delta^{-id} - \delta. \tag{36}$$

Similarly, we have

$$l_n \leq n\delta^{-nd} - \delta. \tag{37}$$

Also, for any $n \geq 1$, we have

$$\left(\frac{2n+1}{2n}\right) \leq \left(\frac{2n+1}{2n}\right)^2 \leq \cdots \leq \left(\frac{2n+1}{2n}\right)^n \leq 2 \tag{38}$$

All-max-shift operations. Following (Yun et al., 2020), to make the interval between l_k are disjoint with each other, the all-max-shift operation $\Omega^l : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ is a self-attention layer defined as follows:

$$\Omega^l(\boldsymbol{L}; c) = \boldsymbol{L} + c\boldsymbol{e}^{(1)}\psi^l(\boldsymbol{L}; 0). \tag{39}$$

The $(1,k)$-th entry of $\Omega^l(\boldsymbol{Z}; c)$ reads

$$\Omega^l(\boldsymbol{L}; c)_{1,k} = L_{1,k} + c\psi^l(\boldsymbol{L}; 0)_k = L_{1,k} + c\max_{j\in\mathcal{A}^l_k} \boldsymbol{u}^T\boldsymbol{L}_j. \tag{40}$$

The main idea of all-max-shift operation is that, in the i-th layer, we will 'replace' the current 'column' by the maximum column within reach of sparse attention pattern $\mathcal{A}^i$. In the next layer, the shifted max column will again be 'replaced' by the new maximum value within reach of the shifted column. After n steps or layers, all the first elements of each column will be replaced by the one in the maximum column, which is the dominated value. The steps within the dominated element are greater than the intervals of the whole l_n. So, for two different inputs $\boldsymbol{L}$, they n entries are distinct, and the requirement 2 in Definition 3.1 satisfied.

Without loss of generality, in contrast with the case of the cycle attention Eq. 12 in the adaptive selective operation, the case of the stacked sampled attention is the same as in Fig. 1d, with $l = 1$.

First layer of all-max-shift. The input of the first all-max-shift operation is $\tilde{\boldsymbol{L}} \in \mathbb{R}^{d\times n}$. Recall that $\boldsymbol{u}^T\tilde{\boldsymbol{L}} = [l_1, \tilde{l}_2, \tilde{l}_3, \dots, \tilde{l}_n]$ and each element is $0 < l_1 < \tilde{l}_2 < \tilde{l}_3 < \dots < \tilde{l}_n < n\delta^{-nd} - \delta$. The last inequality holds as in Eq. 36. Let the output of the first layers be $\boldsymbol{M}^1$. The k-th element in the first row reads

$$M^1_{1,k} := \tilde{L}_{1,k} + 2n^2\delta^{-nd-1} \max_{j\in\mathcal{A}^1_k} \boldsymbol{u}^T\tilde{\boldsymbol{L}}_j = \tilde{L}_{1,k} + 2n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k+1 \mod n}, \tag{41}$$

where with constant value $c = 2n^2\delta^{-nd-1}$ in Eq. 40, and for each column we will have

$$\boldsymbol{u}^T\boldsymbol{M}^1_k = \boldsymbol{u}^T\tilde{\boldsymbol{L}}_k + 2n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k+1 \mod n}, \tag{42}$$

as the first element of $\boldsymbol{u}$ is 1. Next, we see that $\boldsymbol{u}^T\boldsymbol{M}^1_k$ is dominated by the right term $2n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k+1 \mod n}$, which is defined by for any $k, k' \in [n]$,

$$\boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k+1 \mod n} < \boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k'+1 \mod n} \Rightarrow \boldsymbol{u}^T\boldsymbol{M}_k < \boldsymbol{u}^T\boldsymbol{M}_{k'}. \tag{43}$$

This is because the minimum gap between $\boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k+1}$ is δ, and we have

$$\boldsymbol{u}^T\tilde{\boldsymbol{L}}_k < n\delta^{-nd} < 2n^2\delta^{-nd-1}\cdot\delta, \tag{44}$$

so if we have $\boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k+1 \mod n} < \boldsymbol{u}^T\tilde{\boldsymbol{L}}_{k'+1 \mod n}$, it could determine the order $\boldsymbol{u}^T\boldsymbol{M}_k < \boldsymbol{u}^T\boldsymbol{M}_{k'}$, because $\boldsymbol{u}^T\tilde{\boldsymbol{L}}_k$ is within the minimum gap of the right term of Eq. 42, and so cannot change the overall value.

Second layer of all-max-shift. As in the first layer, we define the output of this layer as $\boldsymbol{M}^2$, and the k-th element in the first row reads

$$M^2_{1,k} := M^1_{1,k} + 2n^2\delta^{-nd-1} \max_{j\in\mathcal{A}^2_k} \boldsymbol{u}^T\boldsymbol{M}^2_j = M^1_{1,k} + 2n^2\delta^{-nd-1}\boldsymbol{u}^T\boldsymbol{M}^2_{k+1 \mod n}, \tag{45}$$

so for each column, we have

$$\begin{aligned}
\boldsymbol{u}^T\boldsymbol{M}^2_k &= \boldsymbol{u}^T\boldsymbol{M}^1_k + 2n^2\delta^{-nd-1}\boldsymbol{u}^T\boldsymbol{M}^2_{k+1 \mod n} \\
&= \boldsymbol{u}^T\tilde{\boldsymbol{H}}_k + 2n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+1 \mod n} \\
&\quad + 2n^2\delta^{-nd-1}(\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+1 \mod n} + 2n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+2 \mod n}) \\
&= \boldsymbol{u}^T\tilde{\boldsymbol{H}}_k + 4n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+1 \mod n} + (2n^2\delta^{-nd-1})^2\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+2 \mod n}.
\end{aligned} \tag{46}$$

The last term domains $\boldsymbol{u}^T\boldsymbol{M}^2_k$, because the minimum gap of $\boldsymbol{u}^T\boldsymbol{M}^2_{k+1 \mod n}$ is at least δ, and

$$\begin{aligned}
&\boldsymbol{u}^T\boldsymbol{M}^2_k - (2n^2\delta^{-nd-1})^2\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+2 \mod n} = \boldsymbol{u}^T\tilde{\boldsymbol{H}}_k + 4n^2\delta^{-nd-1}\boldsymbol{u}^T\tilde{\boldsymbol{H}}_{k+1 \mod n} \\
&< (1 + 4n^2\delta^{-nd-1})n\delta^{-nd} \le (1+4n)n^2\delta^{-2nd-1} \le (2n^2\delta^{-nd-1})^2\cdot\delta.
\end{aligned} \tag{47}$$

The last inequality holds due to

$$\left(\frac{1+2n}{2n}\right)^2 \le 2 \Leftrightarrow 1 + 4n \le 4n^2, \tag{48}$$

from Eq. 38.

Repeat all-max-shifts. After all n layers we get $\boldsymbol{M}^n$, and $\boldsymbol{u}^T\boldsymbol{M}_k^n$ is dominated by

$$(2n^2\delta^{-nd-1})^n \max_{j\in\mathcal{A}_k^n} \boldsymbol{u}^T\tilde{\boldsymbol{H}}_j = (2n^2\delta^{-nd-1})^n\tilde{l}_n. \tag{49}$$

Because the remains in $\boldsymbol{u}^T\boldsymbol{M}_k^n$ have strictly upper-bound

$$\boldsymbol{u}^T\boldsymbol{M}_k^n - (2n^2\delta^{-nd-1})^n\tilde{l}_n < \left(\sum_{i=0}^{n-1}\binom{n}{i}(2n^2\delta^{-nd-1})^i\right)n\delta^{-nd} \tag{50}$$

$$\leq \left(\sum_{i=0}^{n-1}\binom{n}{i}(2n)^i\right)(n\delta^{-nd-1})^{n-1}n\delta^{-nd} \tag{51}$$

$$= ((1+2n)^n - (2n)^n)\,(n\delta^{-nd-1})^n\cdot\delta \leq (2n^2\delta^{-nd-1})^n\cdot\delta. \tag{52}$$

The last inequality used $(1+2n)^n - (2n)^n \leq (2n)^n$ from Eq. 38.

Verifying Contextual Mapping. This matches the analysis in §E.2.5 of (Yun et al., 2020). As all $\boldsymbol{u}$ selective-shift operations and all-max operations are bijective, and $\boldsymbol{u}$ map each column (token) of the input to the unique id, the requirement 1 in the Definition 3.1 holds. As $\boldsymbol{u}^T\boldsymbol{M}_k^n$ are all dominated by $(2n^2\delta^{-nd-1})\tilde{l}_n$, and different inputs $\boldsymbol{L}$ have different $\tilde{l}_n$ as $\tilde{l}_n$ is influenced by all $[l_1, l_2, \ldots, l_n]$, not all columns are the same for different inputs $\boldsymbol{L}$, and $\boldsymbol{u}^T$ is the unique mapping. The interval may be written

$$\boldsymbol{u}^T\boldsymbol{M}_k^n \in [(2n^2\delta^{-nd-1})^n\tilde{l}_n, (2n^2\delta^{-nd-1})^n(\tilde{l}_n+\delta)]. \tag{53}$$

The upper bound holds as other terms are less than $(2n^2\delta^{-nd-1})^n\cdot\delta$ in total (not the dominated term). So as we can see the interval for all $\boldsymbol{u}^T\boldsymbol{M}_k^n$ are disjoint for different inputs, and the requirement 2 in the Definition 3.1 holds.