SAMSA: Efficient Transformer for Many Data Modalities

Anonymous authors
Paper under double-blind review

Abstract

The versatility of self-attention mechanism earned transformers great success in almost all data modalities, with limitations on the quadratic complexity and difficulty of training. Efficient transformers, on the other hand, often rely on clever data-modality-dependent construction to get over the quadratic complexity of transformers. This greatly hinders their applications on different data modalities, which is one of the pillars of contemporary foundational modeling. In this paper, we lay the groundwork for efficient foundational modeling by proposing **SAMSA** - SAMpling-Self-Attention, a context-aware linear complexity self-attention mechanism that works well on multiple data modalities. Our mechanism is based on a differentiable sampling without replacement method we discovered. This enables the self-attention module to attend to the most important token set, where the importance is defined by data. Moreover, as differentiability is not needed in inference, the sparse formulation of our method costs little time overhead, further lowering computational costs. In short, SAMSA achieved competitive or even SOTA results on many benchmarks, while being faster in inference, compared to other very specialized models. Against full self-attention, real inference time significantly decreases while performance ranges from negligible degradation to outperformance. We release our source code in supplementary materials.

1 Introduction

Transformers (Vaswani et al., 2017) have been successfully applied to a wide variety of tasks on many data modalities, namely sequence (Vaswani et al., 2017; Radford et al., 2019; Devlin et al., 2019), vision (Dosovitskiy et al., 2021; Carion et al., 2020; Liu et al., 2023a), speech (Gulati et al., 2020; Gong et al., 2021; Koutini et al., 2022), time-series (Liu et al., 2022; Jin et al., 2024), point cloud (Lee et al., 2019; Zhao et al., 2021; Yang et al., 2019), and graph (Yun et al., 2019; Rampášek et al., 2022; Shirzad et al., 2023; Ngo et al., 2023). This can be attributed to the permutation equivariance of transformers, which provides a unique way of encoding structural information through positional encoding (absolute or relative positional enccodings). Ultimately, transformers become the building blocks of contemporary foundational modelings. While transformer can be the one architecture for all data modalities, quadratic complexity that hinders it from processing very long sequences. To mitigate the problem of quadratic complexity, based on the data modality they are working on (mostly sequence), efficient transformers (Beltagy et al., 2020; Ainslie et al., 2020; Zhu et al., 2021; Zaheer et al., 2020; Shirzad et al., 2023) have been developed. This is great for its original usage and some of the mentioned works have found their way to industrial applications. However, it has a major flaw: it only works for the intended data modality and is often hard to adapt to another data modality. For example, it is unreasonable to expect sliding window attention (Zhu et al., 2021) to work on data with non-rigid data structures like point clouds or graphs. Without efficient building blocks working across many data structures, future works toward AGI would be costly in both training and inference.

In this work, we develop towards data modality agnostic efficient transformer architecture to fill the literature gap. Any satisfying architecture should respect the permutation equivariance that full self-attention adheres to for versatility, be both sub-quadratic asymptotic computational cost for very large number of tokens, and be efficient in common cases (i.e. 512 - 1024 tokens) for practical usage. Inspired by the finite-yet-directed nature of human attention, we developed a parallelizable differentiable sampling without replacement algorithm that allows models to learn to choose what set of tokens they should attend to. This parallelization of our

differentiable sampling algorithm enables our model to be trained efficiently while the without-replacement characteristic allows our model to maintain some functional expressivity. Since the result of one attention head is not dependent on the input of any other attention head, different attention heads can use different sampled sets of tokens. This consequentially enlarges the receptive field, which further increases the expressivity of our model.

To demonstrate the effectiveness of our method on many data modalities, we conduct extensive experiments on multilinear and non-multilinear data modalities. For multilinear data modalities, we experimented on sequence data modality via the long-range arena benchmark (Tay et al., 2021) featuring four classification tasks and one retrieval task on these datasets. For non-multilinear data modalities, we experimented on graph and point cloud data modalities: graph classification/regression via peptides dataset (Singh et al., 2015) from Long-Range Graph Benchmark (Dwivedi et al., 2022), object classification on ModelNet40 dataset (Wu et al., 2015), and part segmentation on ShapeNet dataset (Chang et al., 2015). Compared to specialized models with data-modality-specific inductive bias, we achieve competitive results while being much faster than them. Against full self-transformers, the performance degradation was negligible, and in some cases, unexpectedly outperformed full self-attention transformers. Aside from comparative experiments, we also conducted many ablation studies to understand: the efficiency-performance trade-off, regularization effect of sampling mechanisms, and training stability w.r.t number of sampled tokens and the sparse formulation.

In short, our contributions can be summarized as follows:

- Parallelizable differentiable sampling without replacement method and its integration to self-attention mechanism,
- Fast training time and negligible extra cost for inference compared to transformers with negligible performance loss,
- Demonstrated effectiveness on many data modalities, competitive results even against specialized models.

2 Related works

Efficient Transformers. The scalability of transformers is hindered by the quadratic complexity of self-attention modules. Hence, numerous works have delved into sparsity (Beltagy et al., 2020; Ainslie et al., 2020; Zaheer et al., 2020; Shirzad et al., 2023), low-rank projection (Wang et al., 2020), hashing-based method (Kitaev et al., 2020), and kernel-based methods (Choromanski et al., 2021; Katharopoulos et al., 2020) to sacrifice the expressivity of transformers for a sub-quadratic construction. Our review of contemporary efficient transformers is influenced by this survey by (Tay et al., 2022). Linformer (Wang et al., 2020) is based on the low-rank assumption of self-attention matrices, realized via projection matrices. As one of the earliest works in efficient transformers, it has flaws in both scalability and fixed sequence length. Subsequent work like Linear Transformer (Katharopoulos et al., 2020) redesigns the similarity kernel previously as a softmax of dot-product score (or any other kernels) to one made of a linear probability cage. The modification frees them from the recomputation of the multiplication of key-value matrices, making attention computational cost $O(N)$, where N refers to the number of tokens. Without softmax non-linearity, the rank of those attention matrices is bounded by the number of dimensions of tokens, which loses much expressivity of transformer models. Methods like (Beltagy et al., 2020) utilize three choices of fixed-sparsity-patterns: sliding window, dilated sliding window, and global+sliding window. Similarly, (Zaheer et al., 2020) uses a combination of different sparse patterns at one time: random attention, window attention, and global attention. On graphs, (Shirzad et al., 2023) also combines different sparsity patterns but with a constraint on the random-generated one: it has to be an expander graph. The hard-coded sparsity effectively linearized the complexity of transformers and is empirically fast. However, intuitively, no pattern or finite set of patterns fits all problems; therefore, there is also a line of work dedicated to a differentiable sparsity. (Roy et al., 2021) drew a relationship from MIPS problem (Maximum Inner Product Search) and the Nearest-Neighbor Search algorithm, when both the query and key vectors are unit vectors. With their online k-means algorithm, their method attends each query vector to every key vector belonging to the same cluster, thus, bringing the complexity down to $O(n^{1.5})$, with

n is the number of tokens. Practically, however, their method is not fast as it needs a k-means construction and extensive specialized implementation for sparse operators. Another perspective on efficient transformers is making better implementation of vanilla transformers (Rabe & Staats, 2021; Dao et al., 2022; Dao, 2024). (Rabe & Staats, 2021) proposed a neat trick for attention mechanism: chunking. While the mechanism is simple, they have reduced the memory cost from quadratic to sublinear, effectively lifting the computation constraint of GPU memory. (Dao et al., 2022) both reduces the frequency of data transfer between HBM and SRAM within the GPU and derives a memory efficient backward pass. This allows quadratic attention scales to the sequence length of 64000 and beyond. (Dao, 2024) is an incremental work from (Dao et al., 2022) with better work partition and reduction in non-matmul FLOPS.

Continuous Relaxation Discrete operations are much used in decision problems (classification, next-word prediction, plan, ...). However, these operations by nature are not amenable to gradient-based optimization because that operation has gradients everywhere either zero or undefined. For example, the heaviside function for binary choice:

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{1}$$

has gradient undefined at $x = 0$ and zero everywhere else. Since continuous relaxation of discrete operators and algorithms is a complex study and the sampling operator is the core of our efficient transformer formulation, we focus our literature review on differentiable sampling. To continuously relax the sampling operator, the previous literature was developed based on the relaxation of choose, sort, and top-k algorithms (Jang et al., 2017; Maddison et al., 2017; Kool et al., 2019; Xie et al., 2020; Blondel et al., 2020; Petersen et al., 2021). (Jang et al., 2017; Maddison et al., 2017) is the foundational work on parameterization of discrete categorical variable. They introduces the Gumbel-Softmax distribution that approximate categorical distribution exceptionally well. There are some differences between the two works: (Jang et al., 2017) introduces straight through estimator via application of path derivative (Bengio et al., 2013) and (Maddison et al., 2017) takes account of density in their relaxed objective function. From differentiable categorical variable, differentiable sampling with replacement can be obtained trivially by computing these stochastic categorical variable k-times, where k is the number of samples. Differentiable sampling without replacement, however, is much trickier and is an active area of research. (Kool et al., 2019) is one of the earliest work in differentiable sampling without replacement. Their process of sampling without replacement, Gumbel Top-k, samples tokens sequentially with logit normalization, applied in generative text models as stochastic beam search. (Xie et al., 2020) sees the differentiable without replacement in another perspective: optimal transport, where loss function represents how far away is the current solution from the optimal one. (Blondel et al., 2020) is an efficient estimation of sort function differentiation through optimal transport via permutahedron projections. The projection decreases the quadratic computational cost to $O(n \log n)$ forward pass and $O(n)$ backward pass. (Petersen et al., 2021) takes a pre-existing discrete parallelizable sorting/top-k algorithm (e.g. odd-even sorting network, bitonic sorting network, selection network, ...) and relaxes all of the control flows for gradient estimation.

3 Background

3.1 Self-Attention Mechanism

Vanilla Self Attention. The self-attention mechanism is a core component of Transformer models for long-range dependencies. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$ represent a set of tokens of n d-dimensional vectors. The matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ all in $\mathbf{R}^{n \times d_h}$ representing query, key, and value vectors are created via learnable linear transformation of the input token matrix $\mathbf{X}$:

$$\begin{aligned} \mathbf{Q} &= \mathbf{X}W_Q + \mathbf{1}_n b_q^\top, \\ \mathbf{K} &= \mathbf{X}W_K + \mathbf{1}_n b_k^\top, \\ \mathbf{V} &= \mathbf{X}W_V + \mathbf{1}_n b_v^\top. \end{aligned}$$

Then, using the query, key, and value vectors, it creates another set of tokens by aggregating those value vectors based on the dependencies defined through the attention map constructed by query and value vectors:

$$\text{Attention}(\mathbf{X}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V}, \tag{2}$$

This mechanism allows information aggregation of very far tokens and effectively captures complex long-range relationships.

Efficient Quadratic Attention. Computing the given Attention function naively yields a quadratic memory cost due to the need to store attention maps for backpropagation. However, its memory complexity can be reduced into linear complexity by checkpointing (Rabe & Staats, 2021) or algebraic manipulation of derivative function (Dao et al., 2022). Via fused kernel (Dao et al., 2022; Dao, 2024), the computational cost of self-attention is further decreased through fewer memory operations between HBM and SRAM. This allows better parallelization, making quadratic attention scales to the length of 64k and beyond.

3.2 Continuous Relaxation of Top-k Sampling

Top-k Sampling. The top-k sampling operation selects the best set of k objects out of others under a criterion, deterministic or stochastic. The criterion function is a permutation invariant function that maps a set of n vectors to a real number. The following equation describes a mathematical formulation of top-k sampling:

$$\text{Sample}(X, k) = \text{argmax}_S^{C(X,k)}\, \text{criterion}(S), \tag{3}$$

where X is the set of vector to sample from, k is the number of sampled vectors needing to be sampled, and $C(N, k)$ is the set of all k-combinations of set N. Since the argmax operator has no continuity, the derivative of the sampled vectors with respect to the criterion is either zero or undefined, i.e., not useful. Therefore, it is hard to integrate into neural networks. Early work (Papernot & McDaniel, 2018) had to resort to a two-stage optimization. This circumvention, however, introduces additional complexity and worsens performance via non-aligned objectives.

Straight Through Estimator. Before going deeper into the differentiable top-k literature, it is crucial to explain a much simpler case: $k = 1$. To choose one vector from many, (Jang et al., 2017) provides a differentiable approximation of the categorical distribution via Softmax with Gumbel noise added on logits:

$$\text{GumbelSoftmax}(\pi)[i] \triangleq \frac{\exp\left((\pi_i + g_i)/\tau\right)}{\sum_{j=1}^{n} \exp\left((\pi_j + g_j)/\tau\right)},$$

where g_i are independent and identically samples from Gumbel(0,1) distribution, π_i are the unnormalized logits, and τ is the temperature parameter that controls the smoothness of the distribution. By adjusting τ, the Gumbel-Softmax can transition between a one-hot categorical distribution and a softer distribution, enabling gradient-based optimization. The straight-through estimator, however, uses the above soft-categorical function as a differentiable proxy for the hard-categorical function obtained by argmax:

$$\text{ST-GumbelSoftmax}(\pi)[i] \triangleq \begin{cases} 1, & \text{if } i = \text{argmax}(\pi), \\ 0, & \text{otherwise}, \end{cases}$$
$$\partial\text{ST-GumbelSoftmax}/\partial\pi \triangleq \partial\text{GumbelSoftmax}/\partial\pi.$$

A naive way to extend this to differentiable top-k (Kool et al., 2019) is to use Gumbel-Softmax to sample from the unsampled vectors k times sequentially, named Gumbel-Topk. This is not parallelizable and has unreliable gradients; therefore, it cannot be used inside repeating deep learning modules, e.g. transformer layers.

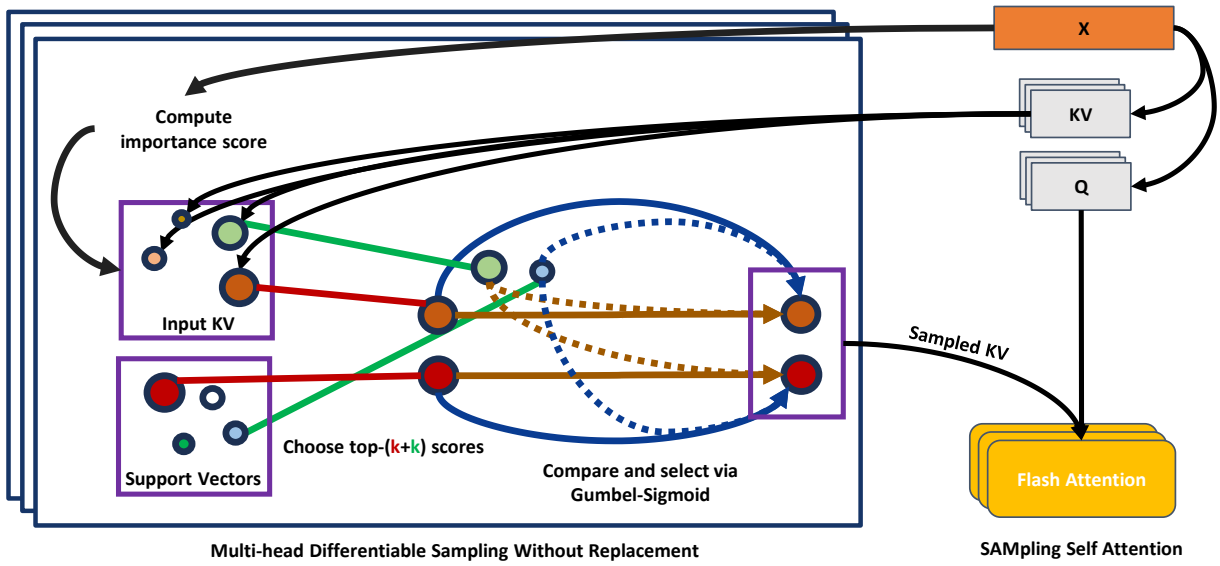


Figure 1: Overview of our proposed model sampling-self-attention module SAMSA. The key-value vectors are selected via top-k importance score computed using tokens' latent. The Gumbel-Sigmoid reparameterization provides gradients to guide the optimization process towards most important key-value pairs of vectors (left). The sampled key-value vectors are then fed into Flash Attention to attend to the query vectors (right).

4 Method

4.1 Overview

Our method is a composition of context-aware sampling without replacement method (Section 4.2) and the integration to self-attention mechanism (Section 4.4). Via reparameterization via Gumbel-Softmax, we devised a parallelizable differentiable sampling without replacement as an innovation from Gumbel-Top-K in existing literature (Kool et al., 2019) - a sequential method. Similar to Linformer (Wang et al., 2020) applying low-rank approximation to key and value vectors, we attend query vectors to the sampled key and value vectors. Inspired by the MoE architecture where each MLP chooses different subsets of tokens to transform (Zhou et al., 2022), our architecture assigns different sets of key-value pairs to different heads. This novel paradigm can be seen as mixture of attention experts, where each expert oversees different sets of tokens. The method is summarized in Algorithm 1.

4.2 Parallelizable Differentiable Top-k Sampling

To be a viable key-value sampler for transformers, that method needs both great parallelizability and reliable gradients for efficient model training. Here, we introduce the two methods that hold the needed properties: trivial differentiable sampling with replacement and our proposed differentiable without replacement method.

With-Replacement Sampling. We define z as the importance score of vectors, representing the vectors' likeliness of being picked. The importance scores of vectors are computed via a neural network: $\text{ImportanceScore}(\mathbf{X}) : \mathbb{R}^{n \times m} \to \mathbb{R}^n$:

$$z \triangleq \text{ImportanceScore}(\mathbf{X}).$$

This simple method conditions the importance score of each token on the entire set of tokens, allowing patterns with complex dependencies to emerge. If we allow the set of sampled vectors to contain duplicates,

Algorithm 1 Sampling-based Transformer Layer

Input: Input token matrix $\mathbf{X}$, sample size k
Output: Updated token matrix $\mathbf{X}$

Transform tokens to query and key-value matrices.
$\widehat{\mathbf{X}} \leftarrow \text{RMSNorm}(\mathbf{X})$
$\mathbf{Q} \leftarrow \widehat{\mathbf{X}}\mathbf{W_q} + \mathbf{1}_n b_q^\top$
$\mathbf{P} \leftarrow \widehat{\mathbf{X}}\mathbf{W_{kv}} + \mathbf{1}_n b_{kv}^\top$
Compute multi-head importance scores
$\mathbf{Z} \leftarrow \text{MultiHeadImportanceScore}(\mathbf{X})$
$\mathbf{Z} \leftarrow \mathbf{Z} + \text{Gumbel}(0,1).\text{sample}()$
Concatenate with 2k support vector paddings
$\mathbf{P} \leftarrow \text{Concatenate}(\mathbf{P}, \mathbf{P}_{\text{supp}})$
$\mathbf{Z} \leftarrow \text{Concatenate}(\mathbf{Z}, \mathbf{Z}_{\text{supp}})$
Sample sets of key and value vectors
$\tilde{\mathbf{P}} \leftarrow \text{MultiHeadSample}(\mathbf{Z}, \mathbf{P}, k)$
$\tilde{\mathbf{K}}, \tilde{\mathbf{V}} \leftarrow \text{Split}(\tilde{\mathbf{P}})$
Attend and transform tokens
$\mathbf{X} \leftarrow \mathbf{X} + \text{FlashAttention}(\mathbf{Q}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}})$
$\widehat{\mathbf{X}} \leftarrow \text{RMSNorm}(\mathbf{X})$
$\mathbf{X} \leftarrow \mathbf{X} + \text{FFN}(\widehat{\mathbf{X}})$
Return X

Gumbel-Top-k algorithm can be parallelized simply by computing everything concurrently as follows:

$$\text{Sample}_1(z, \mathbf{X}, k) \triangleq \text{Concatenate}_{i=1}^{k}(\text{ST-GumbelSoftmax}(z)^\top \mathbf{X}).$$

Without-Replacement Sampling. The weakness of sampling with replacement is that it creates duplicates. These duplicates waste computational resources on the same pieces of information, which harms the expressivity of transformers relying on sampling. This proposed method of without-replacement sampling relies on a novel proxy soft gradient estimator we devised. To begin with, we started with a hypothetical inefficient but parallelizable method: select one set of sampled vectors from sets of all sets of sampled vectors using Gumbel-Softmax (this is the definition in Equation 3). This is equivalent to a continuous relaxation of the discrete brute-force combinatorial optimization algorithm. By defining the likeliness of choosing one set (i.e. the importance score of sets of vectors) as the sum of the importance scores of its elements, we have:

$$z^* \triangleq \text{Concatenate}_{\mathbf{S}}^{C(\mathbb{N}^{\leq n}, k)}(\sum_{i}^{\mathbf{S}} z_i),$$
$$\mathbf{X}^* \triangleq \text{Concatenate}_{\mathbf{S}}^{C(\mathbb{N}^{\leq n}, k)}(\text{vec}(\mathbf{X_S})^\top),$$
$$\text{Sample}_2(z, \mathbf{X}, k) \triangleq \text{ST-GumbelSoftmax}(z^*)^\top \mathbf{X}^*,$$

where $C(N, k)$ is the set of all k-combinations of set N, $\mathbb{N}^{\leq n}$ is the set of all natural numbers less than or equal to n, vec is the matrix vectorization function obtained by stacking the columns of the input matrix on top of one another, $\mathbf{X_S}$ is the matrix of the row vectors with indices in $\mathbf{S}$ of $\mathbf{X}$. This definition of the importance scores of sets has a clear advantage: not needing to enumerate over the entire $C(n, k)$ at inference. Since using all possible combinations to generate one gradient vector for a single optimization step is overkill, an immediate improvement is local search - a sequential algorithm. Intuitively, this allows us to put the optimization steps of local search and gradient descent in the same for loop. For local search, we define the current solution as the set with the highest importance score. The locality of an index set S is then described as the collection of index sets, each differing from S by exactly one element:

$$\text{Current} \triangleq \text{argmax}(z^*) = \text{ArgTopK}(z)$$
$$\text{Local}(\mathbf{S}) \triangleq \{x : x \in C(\mathbf{X}, k) \text{ and } |x - \mathbf{S}| = 1\}$$

Using local search as the base discrete combinatorial optimization algorithm, we compare the current solution against one of its localities, using Gumbel-Softmax reparameterization. The average result from comparing $k(n-k)$ pairs of solutions is given as follows:

$$
\begin{aligned}
i &\triangleq \text{ArgTopK}(z, k), \\
j &\triangleq \text{ArgTopK}(-z, n-k), \\
p &\triangleq J_{k,n-k}, \\
\partial p / \partial z &\triangleq \partial \left(\text{ST-GumbelSigmoid}\left(z_i \mathbf{1}_{n-k}^\top - \mathbf{1}_k z_j^\top \right) \right) / \partial z, \\
\text{Sample}(z, \mathbf{X}, k)[m] &\triangleq \sum_{v=1}^{n-k} \frac{\mathbf{X}[i_m] * p[m, v] + \mathbf{X}[j_v] * (1 - p[m, v])}{n-k},
\end{aligned}
$$

where $J_{m,n}$ is all-ones matrix with m rows and n columns, $\mathbf{1}_k$ are the all-one vectors of length k. The Gumbel Sigmoid function is the Gumbel Softmax when the number of categories is equal to two (details in Appendix D), similar to the relationship between Sigmoid and Softmax functions. To further cut down the optimization cost, we select i as top-k most important and j as top-k second most important index vectors.

4.3 Graph Modeling

The complexity of the graph data structure is needed for the claim of working on many data modalities of ours. Unfortunately, existing literature deemed self-attention alone insufficient for graph and they created very complicated methods: e.g. combination between Message Passing Neural Network (MPNN) and self-attention layers (Rampášek et al., 2022; Ying et al., 2021; Yun et al., 2019). This makes it hard to adapt to other data modalities. We identify the reason transformers need so many modifications just to be barely working on graphs is the inability to process both the nodes and edges information simultaneously. Therefore, we propose Graph-Bridge, a simpler approach treating both nodes and edges as tokens, which can be applied to any self-attention network without complicated constructions.

Formulation. Given a graph $G = (V, E)$ where V represents the set of nodes and E represents the set of edges, the objective is to learn a mapping $\mathcal{F} : G \to \mathbb{R}^d$ that captures the structural properties of G in a d-dimensional latent space. Each node $v_i \in V$ is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^f$, and each edge $e_{ij} = (v_i, v_j) \in E$ is associated with a feature vector $\mathbf{e}_{ij} \in \mathbb{R}^g$.

Positional Encoding. To integrate node and edge information, we introduce a novel graph positional encoding (PE) mechanism. Let $\mathbf{p}_i \in \mathbb{R}^d$ denote the positional encoding of node v_i. We define $\mathbf{p}_i$ as a random vector sampled from a normal distribution:

$$
\mathbf{p}_i \sim \mathcal{N}(0, \text{diag}(\sigma^2)) + \phi(x_i),
$$

where σ^2 is a learnable variance d-dimensional vector and $\phi : \mathbb{R}^f \to \mathbb{R}^d$ is a multi-layer perceptrons (MLPs) to inject node feature into positional encodings. For each edge e_{ij}, its positional encoding $\mathbf{p}_{ij}$ is computed as:

$$
\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j.
$$

While summation is usually used to combine positional encodings, in this case, we use subtraction for edge encoding to preserve the directional information of graph edges.

Node and Edge Transformation. To map node and edge representations into a common latent space, we use two separate MLPs, $\phi_V : \mathbb{R}^d \times \mathbb{R}^f \to \mathbb{R}^d$ and $\phi_E : \mathbb{R}^d \times \mathbb{R}^g \to \mathbb{R}^d$, defined as:

$$
\begin{aligned}
\mathbf{h}_i &= \phi_V(\mathbf{p}_i, \mathbf{x}_i), \\
\mathbf{h}_{ij} &= \phi_E(\mathbf{p}_{ij}, \mathbf{e}_{ij}),
\end{aligned}
$$

where $\mathbf{h}_i$ and $\mathbf{h}_{ij}$ are the transformed representations of node v_i and edge e_{ij}, respectively. These features are then processed by subsequent transformer layers, allowing complex relationships between nodes and edges to be recognized.

Compare to other Graph-PE. This method is simple and efficient to compute: requiring only random feature generation, gathering operation on the randomly generated node PE for edges, and MLP features. This is different to existing graph positional encodings like Random-Walk PE (Ma et al., 2023a) (hard to parallelize) and Laplacian PE (Rampášek et al., 2022) (inherently computationally expensive). While being different, how this method works is surprisingly similar to random-walk PE: each self-attention propagates signals through edge tokens, i.e. we have merged the sequential process of random-walk and multi-layer neural network altogether.

4.4 Integration to Self-Attention Mechanism

While a straightforward integration of our sampling method into the self-attention mechanism is functional, we can achieve significant improvements. The following sections outline our enhancements.

Mixture of Attention Experts. While global attention mechanisms are highly parallelizable and efficient, they limit each token's local receptive field since all tokens attend to the same set of selected tokens. Each attention head operates independently, making self-attention outputs invariant to token permutations within each head. Inspired by MoE (Mixture of Experts) (Zhou et al., 2022), we propose a method where each attention head selects a distinct subset of sampled tokens. This process is parallelizable since the h-sampling processes are independent. Formally, the multi-head importance score $\mathbf{Z}$ is computed by:

$$\mathbf{Z} \triangleq \text{MultiHeadImportanceScore}(\mathbf{X}).$$

And the multi-head sampling function MultiHeadSample as follows:

$$\begin{aligned}\tilde{\mathbf{P}} &\triangleq \text{MultiHeadSample}(\mathbf{Z}, \mathbf{P}, k) \\ &= \text{Stack}_{i=1}^{h}(\text{Sample}(\mathbf{Z}_{\ldots,i}, \mathbf{P}, k)),\end{aligned}$$

where $\mathbf{Z}_{\ldots,i}$ is the i^{th} column vector of $\mathbf{Z}$ matrix. By incorporating this mixture of attention experts, our sampling transformer significantly enhances its local receptive field by a factor of h. For instance, a transformer with four attention heads and a 25% sampling rate can learn a full self-attention pattern if necessary.

Learnable Support Vector Paddings. Simply attending all query vectors to a set of sampled key and value vectors has a limitation: when the number of tokens is less than k (the number of sampled tokens), our algorithm fails. To address this issue, we introduce 2k learnable key and value vectors, each associated with learnable importance scores, which are sampled concurrently with input tokens. The process can be described mathematically as follows:

$$\begin{aligned}\mathbf{P} &\leftarrow \text{Concatenate}(\mathbf{P}, \mathbf{P}_{\text{supp}}), \\ \mathbf{Z} &\leftarrow \text{Concatenate}(\mathbf{Z}, \mathbf{Z}_{\text{supp}})\end{aligned}$$

where $\mathbf{P} : \mathbb{R}^{n \times 2d}$ is the key-value matrix, $\mathbf{Z} : \mathbb{R}^{n \times h}$ is the multi-head importance score matrix, $\mathbf{P}_{\text{supp}} : \mathbb{R}^{\mathbf{2k} \times \mathbf{2d}}$ is the learnable support vector matrix, and $\mathbf{Z}_{\text{supp}} : \mathbb{R}^{\mathbf{2n} \times \mathbf{h}}$ is the learnable multi-head importance score matrix. These vectors and learned importance scores function is much similar to support vectors and Lagrange multipliers in traditional kernel methods, effectively introducing additional decision boundaries within the self-attention mechanism. Consequently, this not only enhances the learning capacity of transformers but also ensures stable training by consistently attending to a fixed number of tokens.

Architecture Details. Our transformer architecture employs the Pre-RMSNorm formulation (Chen et al., 2018; Zhang & Sennrich, 2019), which has been demonstrated to accelerate convergence rates. For the

residual branch, we utilize a zero-initialized learnable scalar multiplier (Bachlechner et al., 2021), with a modification: the learnable scalar multiplier is shared across all network layers. The self-attention modules in our architecture are adapted to incorporate our sampling mechanism, as discussed in detail. The importance score neural network for each self-attention module is implemented as a simple multi-layer perceptron, as the latent representations produced by the preceding transformer stack already exhibit sufficient complexity and expressiveness. Finally, to align with recent advancements in language modeling, we integrate flash attention (Dao et al., 2022; Dao, 2024) into our architecture, which offers higher throughput and reduced memory usage. The full algorithm is summarized in Algorithm 1, which is in the overview section.

4.5 Complexity Analysis

For an input token $\mathbf{X} \in \mathbb{R}^{n \times d}$ with query and key weights $W_Q, W_K \in \mathbb{R}^{d \times d_a}$ and k sampled tokens, the computational costs per 1-head transformer layer are:

- Linear transformations for $\mathbf{Q}, \mathbf{K}, \mathbf{V}$: $O(nd^2)$,
- Calculating token importance scores: $O(nd)$,
- Selecting top-k tokens: $O(n + k)$ forward, $O(n + k^2)$ backward,
- Attention between n queries and top-k keys: $O(nkd_h)$,
- FFN network: $O(nd^2)$.

For large number of tokens ($n \gg d$), the asymptotic time and space complexity of SAMSA layer is $O(n)$.

5 Experiments

5.1 Experiment Design

Our SAMSA models are either using hard/soft sampling indicating whether the straight through estimator is used or not. We have trained **45 models** to measure the effectiveness of our method, both the performance and efficiency-wise of our model, against full self-attention (Flash Attention v2 (Dao, 2024)) and other specialized SOTA models on three data modalities (Sequence, Point Cloud, and Graph) on 4 datasets (Long Range Arena (all five), ModelNet40, ShapeNetPart, and Long Range Graph Benchmark (only Peptides)). To minimize any possible bias when comparing between baselines, we use no data augmentation and identical hyperparameters across different tasks. This also implies we conducted little hyperparameter tuning and mostly focused on the architectural design.

5.2 Performance-Efficiency Trade-offs

We evaluated the performance of our SAMSA model on nine different datasets using six distinct SAMSA model configurations. The detailed results is presented in Table 1.

Trade-offs against Full Self-Attention. At a 50% sampling rate, SAMSA is significantly faster and cheaper to compute than full self-attention, particularly due to the hard formulation, which avoids the need for gradient estimation during inference. Performance-wise, SAMSA consistently outperforms full self-attention, even at very low sampling rates (e.g., Retrieval: 128 / 4k). This superior performance is likely due to the regularization effect introduced by the sampling operator. Early in training, the random selection of tokens acts like DropAttention regularization (Zehui et al., 2019), preventing overfitting by ensuring attention is not overly concentrated on specific tokens. As training progresses, the sampler becomes more effective at identifying important tokens, reducing the regularization effect and allowing for more targeted attention. This dynamic aligns with known phenomena where early-stage regularization accelerates convergence, leading to improved overall performance (Liu et al., 2023b).

Table 1: Performance/Runtime of the full self-attention (Vaswani et al., 2017) compared to SAMSA with three different sampling sizes and two sampling formulations. **Green** and **Red** indicate SAMSA having better or worse statistics compared to full self-attention, respectively. The results of full self-attention in LRA are obtained from (Ma et al., 2023b). It has better performance than what reported in the dataset paper (Tay et al., 2021). The results of full self-attention in LRGB are obtained from the dataset paper (Dwivedi et al., 2022). The results of full self-attention on Shape Classification on ModelNet40 is run by ourselves, with the exact same hyperparameters as other SAMSA models. The results of PartSegmentation on ShapeNetPart is obtained from PointBERT (Yu et al., 2022).

				SAMSA						**Full Self-Attention**
Exp Settings	**Sampling Size**			128		256		512		N/A
	Soft/Hard Sampling			Soft	Hard	Soft	Hard	Soft	Hard	
Sequence	**ListOPS**	L=2048	Acc$^{\uparrow}$	**41.53**	**41.13**	**41.89**	**40.57**	**41.33**	**40.88**	37.11
	Pathfinder	L=1024	Acc$^{\uparrow}$	**81.97**	**71.83**	**80.55**	**69.71**	**79.74**	**72.28**	71.83
	Image	L=1024	Acc$^{\uparrow}$	**48.64**	**48.24**	**48.73**	**47.38**	**48.24**	**46.75**	42.94
	Text	L=2048	Acc$^{\uparrow}$	**64.74**	**65.42**	**65.53**	**65.42**	**65.38**	**65.42**	65.21
	Retrieval	L=4096	Acc$^{\uparrow}$	-	**82.05**	-	**81.89**	-	**81.81**	79.14
Graph	**Pept-func**	L=450	AP$^{\uparrow}$	**72.01**	**72.01**	**71.42**	**72.21**	-	**72.45**	63.26
	Pept-struct	L=450	MAE$^{\downarrow}$	**0.2519**	**0.2458**	-	**0.2515**	-	**0.2468**	0.2529
Point Cloud	**ModelNet40**	L=1024	Acc$^{\uparrow}$	**91.11**	**90.91**	**91.72**	**90.95**	**91.35**	**91.39**	91.07
	ShapeNetPart	L=2500	IoU$^{\uparrow}$	-	**85.74**	-	**85.87**	-	**85.62**	85.1
Inference Speed (1024 tokens)				**1.63×**	**2.17×**	**1.44×**	**1.62×**	**1.08×**	**1.44×**	1×

Trade-offs between Different Sampling Rates. Efficiency-wise, Table 1 shows that SAMSA's inference speed decreases with lower sampling rates and can be twice as fast as full self-attention on an A100. Performance-wise, the table also shows that a sampling rate of 256 tokens per head outperforms both full self-attention and higher sampling rates, suggesting that not all tokens equally impact attention effectiveness. This "sampling nonlinearity" may help filter out redundant tokens. Additionally, SAMSA's performance on easy tasks like LRA-Text and LRA-Image decreases as sampling rates increase, indicating a potentially stronger regularization effect of lower sampling rates.

Trade-offs between Hard and Soft Sampling. Hard-SAMSA is much faster than soft-SAMSA, with a 33% speed advantage at a 50% sampling rate. However, hard-SAMSA underperforms in certain tasks, likely due to discrepancies between the forward and backward passes in Gumbel-Softmax (Jang et al., 2017). Notably, hard-SAMSA's learning curves in the Pathfinder task are volatile, reflecting the adverse effects of the straight-through estimator. Interestingly, earlier navigation through "breakpoints" leads to worse performance, possibly due to flat local minima, as shown by the stair-like learning curves (figure in Appendix E). However, it should be noted that hard-SAMSA performance is not inferior to soft-SAMSA with a counter-example: superior performance on graph data modality.

5.3 Sampling vs. Specialized Models

Aside from comparing against full self-attention, we also compare our model against the SOTA-specialized model on each of our experimented data modalities (sequences, graph, and point cloud).

Sequences. In Table 2, our model has competitive results against many other efficient transformers and the full attention transformer. From the table, it can be seen that MEGA (Ma et al., 2023b) and S4 (Gu et al., 2022) have the highest performance. This is true and expected because recurrent and locality via EMA are very strong inductive bias, making the model only work for sequences. Therefore, it is natural

Table 2: (Long Range Arena) Accuracy on the full suite of long range arena (LRA) tasks. **Bold** indicates best sequence-models while underline indicates best data modality agnostic models.

Method	ListOps	Text	Retrieval	Image	Pathfinder	Modality
BigBird (Zaheer et al., 2020)	36.05	64.02	59.29	40.83	74.87	Sequence
Reformer (Kitaev et al., 2020)	37.27	56.10	53.40	38.07	68.50	Any
Performer (Choromanski et al., 2021)	18.01	65.40	53.82	42.77	77.05	Any
Linformer (Wang et al., 2020)	35.70	53.94	52.27	38.56	76.34	Any
Luna-256 (Ma et al., 2021)	37.98	65.78	79.56	47.86	78.55	Any
Transformer (Vaswani et al., 2017)	37.11	65.21	79.14	42.44	71.83	Any
S4 (Gu et al., 2022)	**88.65**	76.02	87.09	86.09	86.05	Sequence
MEGA (Ma et al., 2023b)	63.14	**90.43**	**91.25**	**90.44**	**96.01**	Sequence
hard-SAMSA (Ours)	41.12	65.42	81.89	48.24	72.27	Any
soft-SAMSA (Ours)	41.88	65.53	82.05	48.73	81.97	Any

Table 3: Experimental results on Peptides func and Peptides struct. The results of methods other than ours are taken from their respective works and LRGB (Dwivedi et al., 2022). We report the means and standard deviations of 4 random seeds.

Method	Peptides Struct MAE ↓	Peptides Func AP ↑
GCN (Kipf & Welling, 2017)	0.3496 ± 0.0013	0.5930 ± 0.0023
GINE (Xu et al., 2019)	0.6346 ± 0.0071	0.5498 ± 0.0079
GatedGCN (Bresson & Laurent, 2017)	0.3420 ± 0.0013	0.5864 ± 0.0077
GatedGCN + RWSE (Bresson & Laurent, 2017)	0.3357 ± 0.0006	0.6069 ± 0.0035
Drew-GCN + LapPE (Gutteridge et al., 2023)	0.2536 ± 0.0015	0.7150 ± 0.0044
GraphGPS + LapPE (Rampášek et al., 2023)	0.2500 ± 0.0005	0.6535 ± 0.0041
GRIT (Ma et al., 2023a)	0.2460 ± 0.0012	0.6988 ± 0.0082
Graph VIT (He et al., 2023)	**0.2449 ± 0.0016**	0.6942 ± 0.0075
GraphMLP Mixer (He et al., 2023)	0.2475 ± 0.0015	0.6921 ± 0.0054
MPNN + VN + RWSE (Cai et al., 2023)	0.2529 ± 0.0009	0.6685 ± 0.0062
24-layer hard-SAMSA (ours)	0.2458 ± 0.0013	**0.7221 ± 0.0042**
24-layer soft-SAMSA (ours)	0.2519 ± 0.0011	0.7171 ± 0.0051

that our method, which is designed for multiple-data modalities, cannot outperform methods like those. Against weaker to no inductive bias like the rest, we achieved better performance on every task except text classification (losing Luna-256 by 0.25%).

Graph. In Table 3, our model has outperformed most methods in the two tasks we tested. This shows the efficacy of our newly proposed Graph-PE. Note that, our method's learnable positional encoding is computed cheaply, and how cheap the computation is best represented in Table 4. The table shows that our entire model computes everything even before other methods finish their positional encoding computation. It also shows the efficacy of the SAMSA model as a very deep network, which is also different from traditional graph learning methods which cannot extend beyond 8 layers due to over-smoothing.

Table 4: Runtime of Graph Positional Encoding on one epoch on A100 compared to our 24-layer SAMSA model. The runtime of LapPE and RWPE are taken from (Dwivedi et al., 2022).

Method	LapPE	RWPE	SAMSA (whole model)
Peptides Struct	73s	53s	**35s**

Table 5: Inference time (in seconds) of the networks for ModelNet40 classification test split in 1 A100 and 8 CPUs with a batch size of 32. The results of PointNet and DGCNN are taken from (Kaba et al., 2023).

Method	PointNet	DGCNN	hard-SAMSA (ours)	soft-SAMSA (ours)
Inference Time	18s	23s	**<1s**	**1s**

Table 6: Experimental results on ModelNet40 dataset (Wu et al., 2015) and part segmentation results on ShapeNetPart dataset (Chang et al., 2015) against point-cloud specialized baselines.

Method	**ModelNet40**		**ShapeNetPart**	
	mAcc$^{\uparrow}$	**OA**$^{\uparrow}$	**c. IoU**$^{\uparrow}$	**i. IoU**$^{\uparrow}$
PointNet (Charles et al., 2017)	86.2	89.2	80.4	83.7
Set Transformer (Lee et al., 2019)	-	90.4	-	-
PointNet++ (Qi et al., 2017)	-	91.9	81.9	85.1
SpecGCN (Wang et al., 2018)	-	92.1	-	-
PointCNN (Li et al., 2018)	88.1	92.2	84.6	86.1
DGCNN (Wang et al., 2019)	90.2	92.2	82.3	85.1
PointWeb (Zhao et al., 2019)	89.4	92.3	-	-
SpiderCNN (Xu et al., 2018)	-	92.4	81.7	85.3
PointConv (Wu et al., 2019)	-	92.5	82.8	85.7
Point2Sequence (Liu et al., 2019)	90.4	92.6	-	85.2
KPConv (Thomas et al., 2019)	-	92.9	**85.1**	86.4
InterpCNN (Mao et al., 2019)	-	93.0	84.0	86.3
Point Transformer (Zhao et al., 2021)	**90.6**	**93.7**	83.7	**86.6**
hard-SAMSA (Ours)	88.9	91.3	83.8	85.8
soft-SAMSA (Ours)	88.7	91.7	-	-

Point Cloud. Performance-wise, Table 6 shows that SAMSA has competitive results against other point cloud models. This is mainly because our model does not include any data-modality-specific inductive bias (like the usage of kNN for locality heuristic); therefore, it is susceptible to overfitting in scenarios where data is not abundant. Our method is also much faster than others, up to 18× faster than PointNet (which is the most small, simple, efficient point cloud model), as shown in Table 5.

6 Conclusion

SAMSA (SAMpling-based Self-Attention) mechanism addresses the limitations of traditional transformers by reducing the quadratic complexity typically associated with self-attention. Unlike other efficient transformers that are often data-modality-specific, SAMSA provides a versatile, context-aware linear complexity solution that can be applied across various data modalities. By employing a differentiable sampling method to focus on the most important tokens, SAMSA achieves competitive or even state-of-the-art results on multiple benchmarks. Additionally, SAMSA's sparse formulation during inference minimizes computational costs, offering faster inference times with little in performance, and in some cases, even outperforming full self-attention transformers.

Limitations. Despite the promising results achieved with SAMSA, there are certain limitations on the misaligned gradients of hard sampling formulation, which makes our model hard to converge in more difficult optimization problems (Pathfinder). This misaligned gradient issue in hard sampling methods is mostly from the misestimation of gradients of the non-chosen path. We believe that future work should focus more precise gradient estimation would result in an inference-fast model with much better accuracy.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 268–284, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.19. URL https://aclanthology.org/2020.emnlp-main.19.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: fast convergence at large depth. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/bachlechner21a.html.

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

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *CoRR*, abs/1711.07553, 2017. URL http://arxiv.org/abs/1711.07553.

Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between mpnn and graph transformer. *arXiv preprint arXiv:2301.11956*, 2023.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, pp. 213–229, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8.

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.

R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017. doi: 10.1109/CVPR.2017.16.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1008. URL https://aclanthology.org/P18-1008.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=in7XC5RcjEn.

Yuan Gong, Yu-An Chung, and James Glass. AST: Audio Spectrogram Transformer. In *Proc. Interspeech 2021*, pp. 571–575, 2021. doi: 10.21437/Interspeech.2021-698.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Anmol Gulati, Chung-Cheng Chiu, James Qin, Jiahui Yu, Niki Parmar, Ruoming Pang, Shibo Wang, Wei Han, Yonghui Wu, Yu Zhang, and Zhengdong Zhang (eds.). *Conformer: Convolution-augmented Transformer for Speech Recognition*, 2020.

Benjamin Gutteridge, Xiaowen Dong, Michael Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay, 2023.

Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs, 2023.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Unb5CVPtae.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

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.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, 2019.

Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. In *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, pp. 2753–2757. ISCA, 2022. doi: 10.21437/Interspeech.2022-227. URL https://doi.org/10.21437/Interspeech.2022-227.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

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 Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf.

Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023a.

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 Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33018778. URL https://doi.org/10.1609/aaai.v33i01.33018778.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=ucNDIDRNjjv.

Zhuang Liu, Zhiqiu Xu, Joseph Jin, Zhiqiang Shen, and Trevor Darrell. Dropout reduces underfitting. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, K. Dokania, Mark Coates, Philip H.S. Torr, and Ser-Nam Lim. Graph Inductive Biases in Transformers without Message Passing. In *Proc. Int. Conf. Mach. Learn.*, 2023a.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453, 2021.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: Moving average equipped gated attention. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=qNLe3iq2El.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1578–1587, 2019. doi: 10.1109/ICCV.2019.00166.

Nikita Nangia and Samuel Bowman. ListOps: A diagnostic dataset for latent tree learning. In Silvio Ricardo Cordeiro, Shereen Oraby, Umashanthi Pavalanathan, and Kyeongmin Rim (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 92–99, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-4013. URL https://aclanthology.org/N18-4013.

Nhat Khang Ngo, Truong Son Hy, and Risi Kondor. Multiresolution graph transformers and wavelet positional encoding for learning long-range and hierarchical structures. *The Journal of Chemical Physics*, 159(3): 034109, 07 2023. ISSN 0021-9606. doi: 10.1063/5.0152833. URL https://doi.org/10.1063/5.0152833.

Nicolas Papernot and Patrick D. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *CoRR*, abs/1803.04765, 2018. URL http://arxiv.org/abs/1803.04765.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Differentiable sorting networks for scalable sorting and ranking supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8546–8555. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/petersen21a.html.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

Markus N. Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory, 2021.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. The ACL Anthology network corpus. In Min-Yen Kan and Simone Teufel (eds.), *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries (NLPIR4DL)*, pp. 54–61, Suntec City, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/W09-3607.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. *Advances in Neural Information Processing Systems*, 35, 2022.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer, 2023.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://aclanthology.org/2021.tacl-1.4.

P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The princeton shape benchmark. In *Proceedings Shape Modeling Applications, 2004.*, pp. 167–178, 2004. doi: 10.1109/SMI.2004.1314504.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Sandeep Singh, Kumardeep Chaudhary, Sandeep Kumar Dhanda, Sherry Bhalla, Salman Sadullah Usmani, Ankur Gautam, Abhishek Tuknait, Piyush Agrawal, Deepika Mathur, and Gajendra P.S. Raghava. SATPdb: a database of structurally annotated therapeutic peptides. *Nucleic Acids Research*, 44(D1):D1119–D1126, 11 2015. ISSN 0305-1048. doi: 10.1093/nar/gkv1114. URL https://doi.org/10.1093/nar/gkv1114.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6410–6419, 2019. doi: 10.1109/ICCV.2019.00651.

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, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 52–66, 2018.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), oct 2019. ISSN 0730-0301. doi: 10.1145/3326362. URL https://doi.org/10.1145/3326362.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9613–9622, 2019. doi: 10.1109/CVPR.2019.00985.

Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society. doi: 10.1109/CVPR.2015.7298801. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298801.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20520–20531. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ec24a54d62ce57ba93a531b460fa8d18-Paper.pdf.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, pp. 90–105, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-01236-6. doi: 10.1007/978-3-030-01237-3_6. URL https://doi.org/10.1007/978-3-030-01237-3_6.

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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3318–3327, 2019. doi: 10.1109/CVPR.2019.00344.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=OeWooOxFwDa.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. *Graph Transformer Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Manzil Zaheer, Guru Guruganesh, Kumar 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 H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Lin Zehui, Pengfei Liu, Luyao Huang, Junkun Chen, Xipeng Qiu, and Xuanjing Huang. Dropattention: A regularization method for fully-connected self-attention networks, 2019. URL https://arxiv.org/abs/1907.11065.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5560–5568, 2019. doi: 10.1109/CVPR.2019.00571.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16239–16248, 2021. doi: 10.1109/ICCV48922.2021.01595.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*,

volume 35, pp. 7103–7114. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2f00ecd787b432c1d36f3de9800728eb-Paper-Conference.pdf.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17723–17736. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9425be43ba92c2b4454ca7bf602efad8-Paper.pdf.

A Datasets

ModelNet40. The ModelNet40 dataset (Wu et al., 2015) consists of 12,311 pre-aligned shapes divided into 40 classes, where the train and test sets consist of 9,843 instances and 2,468 instances respectively. ModelNet40 is the pioneer large-scale 3D CAD dataset. Unlike previous CAD datasets (Shilane et al., 2004), ModelNet40 is the pioneer large-scale dataset that is diverse in terms of both class and samples per class.

ShapeNetPart. The ShapeNetPart dataset consists of 16,881 pre-aligned 3D shapes from 16 categories and is a part of a larger dataset: ShapeNetCore (51,300 3D models) (Chang et al., 2015). The 3D shapes in the ShapeNetPart dataset are annotated with 50 segmentation parts in total representing virtual real-world 3D semantic models. ShapeNet provides a diverse variety of shape annotations and corresponding shapes. The full ShapeNet dataset a is multitude containing upright and front orientation vectors, parts and keypoints, shape symmetries, but we only account for the part-segmenting task in our work.

Long Range Arena. The Long Range Arena (LRA) (Tay et al., 2021) is a composition benchmark of 5 tasks: ListOps (Nangia & Bowman, 2018), ImDB review (Maas et al., 2011), ACL Anthology Network (Radev et al., 2009), Grayscaled CIFAR-10 (Krizhevsky & Hinton, 2009), and Pathfinder (Linsley et al., 2018). These five tasks are all classification and they feature very long sequences: ListOps (2,048 tokens), ImDB review (1,024 tokens), ACL Anthology Network (4,096 tokens), Grayscaled CIFAR-10 (1,024 tokens), and Pathfinder (1,024 tokens). All five tasks involves tackling long-ranged data structures and manifold categorizations, challenging networks's generalization powers as well as memory and time efficiencies.

Long Range Graph Benchmark. The Long Range Graph Benchmark (LRGB) (Dwivedi et al., 2022) is a graph composition of five datasets. But in our work, we only consider two peptide molecular tasks: Peptides-struct and Peptides-func. The chemical structures and features of these peptides are represented as graph of around 155 nodes and diameter of around 57. These challenges are suitable for evaluating long range receptive fields of graph-based deep learning networks. Peptides-struct and Peptides-func are much similar, except the former is a graph regression task while the other is a graph classification task.

B Training Strategies and Data Preprocessing

We use the `pytorch` framework (Paszke et al., 2019) to conduct our empirical experiments.

Here is the list of the common hyperparameters:

- Optimizer: AdamW (Loshchilov & Hutter, 2019)
- Batch size: 32
- Learning rate: 0.001
- β_1, β_2: 0.9, 0.999
- Learning scheduler: Cosine Annealing
- Learning rate warm-up: Linear Warmup-2000 steps
- Gradient Clipping: Max Gradient Norm (2.0)

Our training phase details can be visited in Table 7.

As for the input preprocessing, our works are as follows:

- **Long Range Arena:** We simply use one-hot encoding vectors as inputs.
- **Long Range Graph Benchmark:** We input the whole graph, with both node and edge as tokens.

Table 7: Training hyperparameter details.

Task	Dataset	Learning Rate	Gradient Clipping	Weight Decay	No. of Epochs
Point Cloud	ModelNet40	0.002	2.0	0.1	200/600
	ShapeNetPart	0.002	2.0	0.1	600
Sequence	ListOPS	0.001	2.0	0.01	50
	Text	0.001	2.0	0.01	50
	Retrieval	0.001	2.0	0.01	50
	Image	0.001	2.0	0.01	50
	Pathfinder	0.001	2.0	0.01	50
Graph	Peptides-struct	0.001	2.0	0.01	240
	Peptides-func	0.0004	2.0	0.01	160

- **ModelNet40:** Following previous work (Charles et al., 2017), we use farthest point sampling to sample 1024 initial points with normals.

- **ShapeNetPart:** We operate on the initial 2,500 point-normal without any change to the dataset.

All of the classification task training processes utilize Cross Entropy Loss, without label smoothing normalization. For Peptides-struct, we use SmoothL1Loss for optimization as the measuring metric is Mean Absolute Error (MAE). For Peptides-Func, we use threshold = 0.5 for Average Precision computation, implying we do not tune threshold parameter based on training data, aligning with previous methods.

C Architectural Specification

Table 8 shows our model's architectural hyperparameters, for reproducibility purposes.

Table 8: Architectural details.

Dataset	$\mathbf{n_{depth}}$	$\mathbf{d_{model}}$	$\mathbf{d_{FFN}}$	$\mathbf{n_{heads}}$	$\mathbf{p_{dropout}}$	$\mathbf{p_{droppath}}$
ModelNet40	6	256	768	8	0.1	0.1
ShapeNetPart	8	256	768	8	0.1	0.1
LRA-ListOPS	6	256	768	8	0.1	0.0
LRA-Pathfinder	6	128	512	8	0.3	0.0
LRA-Text	4	128	512	8	0.3	0.0
LRA-Image	8	128	512	8	0.3	0.0
LRA-Retrieval	6	128	512	8	0.3	0.0
LRGB-Peptides-Func	24	128	512	8	0.3	0.2
LRGB-Peptides-Struct	24	128	512	8	0.3	0.2

LRA Tasks use Sinusoid-1D positional encodings.

D Miscellaneous

ST-Gumbel-Sigmoid's Formula:

$$\text{GumbelSigmoid}(\pi)[i] \triangleq \frac{1}{1+\exp\left((\pi_j+g_j)/\tau\right)},$$
$$\text{ST-GumbelSigmoid}(\pi) \triangleq 1\begin{cases}1, & \text{if } \pi \leq 0,\\ 0, & \text{otherwise,}\end{cases}$$
$$\partial\text{ST-GumbelSigmoid}/\partial\pi \triangleq \partial\text{GumbelSigmoid}/\partial\pi,$$

where g_i are independent and identically samples from Gumbel(0,1) distribution, π_i are the unnormalized logits, and τ is the temperature parameter that controls the smoothness of the distribution.

Implementation of our Graph-Bridge PE in `PyTorch`:

```
class GraphBridge(nn.Module):
### Generate Randomized Absolute Positional Encoding for Graphs.
### Input: node_features (b, n_nodes, d_nodes), edge_features (b, n_edges, d_edges),
↪  connection (b, n_edges, 2)

    def __init__(self, d_model, d_random_feature):
        super().__init__()
        self.d_model = d_model
        self.d_random_features = d_random_feature
        self.hadamard_coeffs_l = nn.Parameter(torch.ones(1, 1, self.d_random_features),
        ↪  requires_grad=True)
        self.hadamard_coeffs_r = nn.Parameter(torch.ones(1, 1, self.d_random_features),
        ↪  requires_grad=True)
        self.mlp_node, self.mlp_edge, self.mlp_node_pe = MLP(...), MLP(...), MLP(...)

    def forward(self, node_features, edge_features, mask_node, mask_edge, connection,
    ↪  **kwargs):
        b, n_nodes, _ = node_features.shape
        node_random_positional_l = torch.randn(b, n_nodes, self.d_random_features,
        ↪  device=node_features.device).unsqueeze(2) *
        ↪  F.softplus(self.hadamard_coeffs_l)
        node_random_positional_r = torch.randn(b, n_nodes, self.d_random_features,
        ↪  device=node_features.device).unsqueeze(2) *
        ↪  F.softplus(self.hadamard_coeffs_r)
        node_random_positional = torch.cat([node_random_positional_l,
        ↪  node_random_positional_r], dim=2)
        node_features_add_pe =
        ↪  self.mlp_node_pe(node_features).reshape(node_features.shape[0],
        ↪  node_features.shape[1], 2, self.d_random_features)
        node_random_positional = node_random_positional + node_features_add_pe
        connection = connection.unsqueeze(-1).expand(-1, -1, -1, self.d_random_features)
        edge_pe = torch.gather(F.pad(node_random_positional, (0, 0, 0, 0, 0,
        ↪  edge_features.shape[1])), dim=1, index=connection)
        edge_pe = torch.subtract(edge_pe[:, :, 0, :], edge_pe[:, :, 1, :])
        node_features = self.mlp_node(torch.cat([node_features,
        ↪  node_random_positional.reshape(b, n_nodes, 2 * self.d_random_features)],
        ↪  dim=2))
        edge_features = self.mlp_edge(torch.cat([edge_features, edge_pe], dim=2))

        return {
            'x': torch.cat([node_features, edge_features], dim=1),
```

```
        'mask': torch.cat([mask_node, mask_edge], dim=1),
        'n_nodes': n_nodes,
        'n_edges': edge_features.shape[1]
    }
```

E Learning Curves of Our Models

Figure 2, 3, 4 show the learning curve of SAMSA in various settings, providing for future understanding of how our model learned. All of the lines are plotted with EMA-5 smoothing.

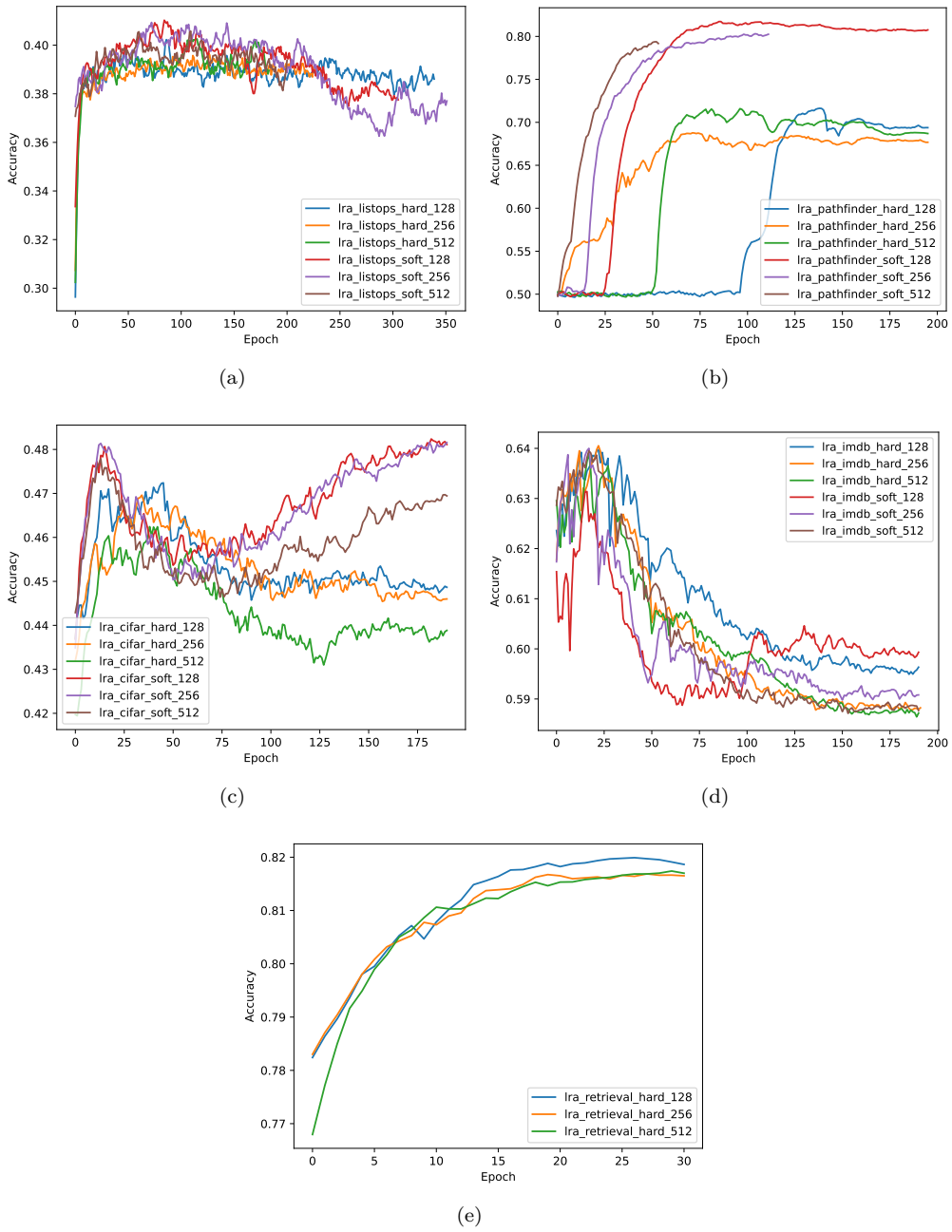
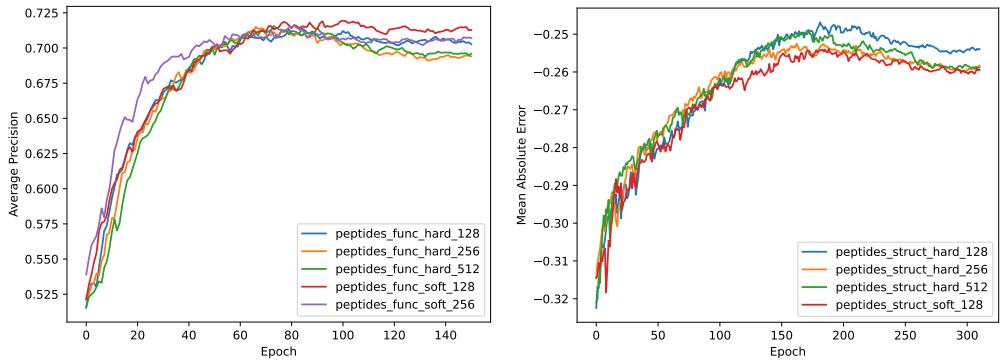
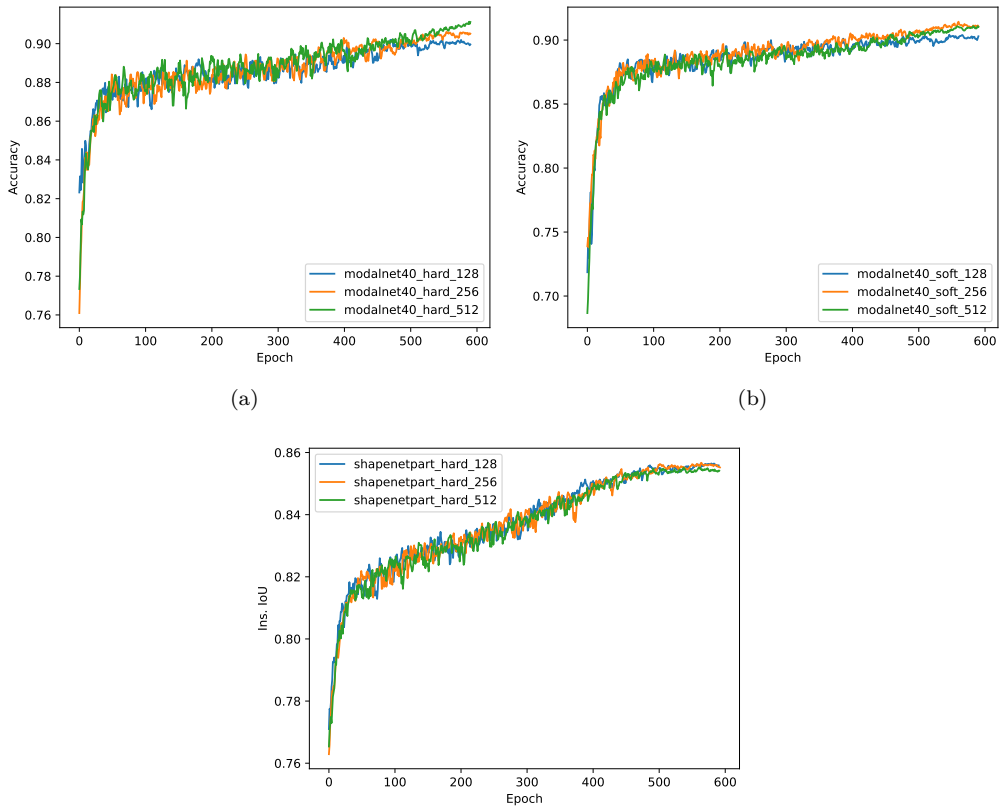


Figure 2: Learning Curves of SAMSA models in Sequence Tasks



(a) (b)

Figure 3: Learning Curves of SAMSA models in Graph Tasks



(a) (b)

(c)

Figure 4: Learning Curves of SAMSA models in Point Cloud Tasks