Recasting Self-Attention with Holographic Reduced Representations

Mohammad Mahmudul Alam [1] **Edward Raff** [1 2 3] **Stella Biderman** [2 3 4] **Tim Oates** [1] **James Holt** [2]

Abstract

In recent years, self-attention has become the dominant paradigm for sequence modeling in a variety of domains. However, in domains with very long sequence lengths the $\mathcal{O}(T^2)$ memory and $\mathcal{O}(T^2 H)$ compute costs can make using transformers infeasible. Motivated by problems in malware detection, where sequence lengths of $T \geq 100,000$ are a roadblock to deep learning, we re-cast self-attention using the neuro-symbolic approach of Holographic Reduced Representations (HRR). In doing so we perform the same high-level strategy of the standard self-attention: a set of queries matching against a set of keys, and returning a weighted response of the values for each key. Implemented as a "Hrrformer" we obtain several benefits including $\mathcal{O}(TH \log H)$ time complexity, $\mathcal{O}(TH)$ space complexity, and convergence in $10\times$ fewer epochs. Nevertheless, the Hrrformer achieves near state-of-the-art accuracy on LRA benchmarks and we are able to learn with just a single layer. Combined, these benefits make our Hrrformer the first viable Transformer for such long malware classification sequences and up to $280\times$ faster to train on the Long Range Arena benchmark. Code is available at https://github.com/NeuromorphicComputationResearchProgram/Hrrformer

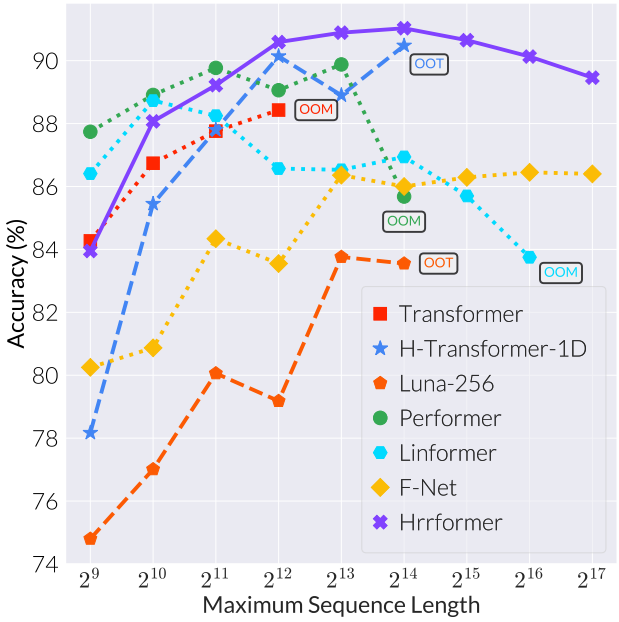

Figure 1: Our primary result, comparison of our Hrrformer with other self-attention models in EMBER malware classification dataset. Most prior methods fail early by running Out Of Memory (OOM) or Time (OOT). Hrrformer is presented in a *solid* line and achieves the best accuracy, scales to longer sequences. The two prior best models according to the Long Range Arena, H-Transformer-1D and Luna-256, are in the *dashed* lines, and do not perform as well as the LRA would have indicated in speed or accuracy. The rest of the models are in the *dotted* line.

1. Introduction

Self-attention has risen to prominence due to the development of transformers (Vaswani et al., 2017) and their recent successes in machine translation, large language modeling, and computer vision applications. The fundamental construction of self-attention includes a triplet of "queries, keys, and values", where the response is a weighted average over the values based on the query-key interactions. This results in a quadratic memory and computational complexity, that has inhibited the use of Transformers to those without significant GPU infrastructure and prevented applications to longer sequences. Ever since, a myriad of approaches has been proposed to approximate the self-attention mechanism, with the vast majority trading some amount of accuracy for speed or memory use. The "market" of self-attention strategies currently offers various trade-offs in the total package of speed, memory use, and accuracy.

[1]Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD, USA [2]Laboratory for Physical Sciences, College Park, MD, USA [3]Booz Allen Hamilton, McLean, VA, USA [4]EleutherAI. Correspondence to: Edward Raff <Raff_Edward@bah.com>, Tim Oates <oates@cs.umbc.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

We test our method in two settings: using the Long Range Arena (LRA) to compare with prior approaches and a real-world task in malware detection. These results show several benefits to the Hrrformer: it is near state-of-the-art in terms of accuracy, and one of only two methods to improve upon the original Transformer for *all* tasks in the LRA. The Hrrformer sets a new benchmark for state-of-the-art speed and memory use, processing $28\times$ more samples/second and using 79.15% less memory than the best prior art for each respective metric. The Hrrformer converges in $10\times$ fewer epochs and is effective with just a single layer. Combined this makes the Hrrformer up to $280\times$ times faster to train. On our malware classification task, we find that the relative accuracies of Transformer models change from the LRA benchmark, but that our Hrrformer still obtains the best accuracy and scales the best with sequence length up to $T = 131,072$, as demonstrated in Figure 1.

The remainder of our manuscript is organized as follows. Work related to our own, as well as adjacent techniques beyond our study's scope, is reviewed in section 2. The recasting of attention in our Hrrformer is a simple procedure demonstrated in section 3, which redefines the *Attention* function using HRR, and multi-headed self-attention then continues as normal. We then demonstrate these benefits in section 4, showing Hrrformer is consistently one of the best methods with respect to accuracy and considerably faster thanks to reduced memory usage, the number of layers, and epochs needed to converge. In section 5 we draw conclusions from out work.

2. Related Works

Since the introduction of the Self-Attention mechanism and the transformer architecture, considerable research has occurred to mitigate its computational burdens. Though not explicit in much of the current literature, many of these approaches resemble strategies for improving Support Vector Machines that have similar complexity. This includes projection (Kaban, 2015) to a lower dimension (Wang et al., 2020), finding/creating sparse structure in the correlations (Wang et al., 2014) by (Kitaev et al., 2020; Child et al., 2019; Tay et al., 2020b; Beltagy et al., 2020; Zaheer et al., 2020), using randomized features (Rahimi & Recht, 2007; Sinha & Duchi, 2016) by (Choromanski et al., 2020), factorized or budgeted representations (Si et al., 2016; Wang et al., 2010) by (Xiong et al., 2021; Ma et al., 2021), and creating simplified linear approximations (Wang et al., 2011; Kantchelian et al., 2014) by (Katharopoulos et al., 2020). Other more differentiated approaches include the hierarchical decomposition of the correlations (by (Zhu & Soricut, 2021)), and approaches that replace self-attention entirely with alternative "mixing" strategies (Tay et al., 2020a; Lee-Thorp et al., 2021). To the best of our knowledge, ours is the first work that attempts to re-create the same logic of self-attention with the HRR.

Among these prior methods, we note that F-Net (Lee-Thorp et al., 2021) is the most closely related as both F-Net and HRR rely upon the Fast Fourier Transform (FFT) as a fundamental building block. While F-Net does not approximate self-attention so much as replace it with an alternative "mixing" procedure, we include it due to its relevance in using the FFT. Our results will show significant improvement over F-Net, highlighting the value of a neuro-symbolic approach to reconstructing the same logic as opposed to using the FFT as a generic differentiable mixing strategy.

The HRR has seen successful use in cognitive science research (Jones & Mewhort, 2007; Blouw & Eliasmith, 2013; Stewart & Eliasmith, 2014; Blouw et al., 2016; Eliasmith et al., 2012; Singh & Eliasmith, 2006; Bekolay et al., 2014), but comparatively little application in modern deep learning. The symbolic properties have been previously used in knowledge graphs (Nickel et al., 2016) and multi-label classification (Ganesan et al., 2021). There is limited use of HRRs for sequential modeling. (Plate, 1992) proposed an HRR-based Recurrent Neural Network (RNN), while other work has used complex numbers inspired by HRRs but not actually used the corresponding operations (Danihelka et al., 2016). An older alternative to the HRR, the Tensor Product Representation (TPR) (Smolensky, 1990) has been used to endow associative memories (Le et al., 2020) and RNNs with enhanced functionality (Huang et al., 2018; Schlag & Schmidhuber, 2018). Compared to these prior works, we are re-casting the logic into HRRs, rather than augmenting the logic. However, we slightly abuse the assumptions of HRRs to make our method work. A strategic design allows us to effectively remove additionally created noise via the softmax function. In addition, the TPR's complexity is exponential in the number of sequential bindings, making it a poor choice for tackling the scaling problems of self-attention.

Other recent approaches to sequential modeling such as Legendre Memory Units (Voelker et al., 2019), IGLOO (Sourkov, 2018), and State Space Models(Gu et al., 2022; Goel et al., 2022; Gu et al., 2021; 2020) are highly promising. We consider these, along with RNNs, beyond the scope of our work. Our goal is to explore the value of recasting self-attention within the neuro-symbolic framework of HRR. As such, other sequence modeling approaches are out of scope.

The need for both less memory and extension to very long sequences is also important in malware detection. Processing malware from raw bytes has been found to be one of the most robust feature types in the face of common malware obfuscations (Aghakhani et al., 2020), but simple n-gram based features have been maligned for being unable

to learn complex sequential information when executable can be tens of kilobytes on the small side and hundreds of megabytes on the larger side (Kephart et al., 1995; Abou-Assaleh et al., 2004; Kolter & Maloof, 2006; Raff et al., 2019; Zak et al., 2017). Given that a maximum $T = 200M$ is realistic, many strategies to handle such sequence lengths have been developed. These include attempts to create "images" from malware (Nataraj et al., 2011; Liu & Wang, 2016), using compression algorithms as a similarity metric (Li et al., 2004; Walenstein & Lakhotia, 2007; Borbely, 2015; S. Resende et al., 2019; Menéndez et al., 2019; Raff & Nicholas, 2017; Raff et al., 2020), and attempts to scale 1D-convolutional networks over raw bytes (Krčál et al., 2018; Raff et al., 2018; 2021).

We will use the Ember (Anderson & Roth, 2018) dataset for malware detection as a real-world test of our new self-attention for processing long sequences. It has been observed empirically that "best practices" developed in the machine learning, computer vision, and natural language processing communities do not always transfer to this kind of data. For example, this phenomenon has been observed with CNNs (Raff et al., 2018) and Transformers for malicious URL detection (Rudd & Abdallah, 2020). Most recently, (Rudd et al., 2022) attempted to apply Transformers to raw byte prediction and had to use a chunked attention that limits the attention window (Sukhbaatar et al., 2019). Using Hrrformer we show much longer sequence processing than this prior work, while simultaneously demonstrating that our method generalizes to a domain that is notorious for a lack of transfer. This increases our confidence in the effectiveness of our method. Notably, the two current state-of-the-art Transformers as measured by the Long Range Arena (LRA) (Tay et al., 2020c) benchmarks do not pass this test, performing considerably worse on the malware task.

3. Attention with Holographic Reduced Representations

The HRR operation allows assigning abstract concepts to numerical vectors, and performing *binding* ($\circledast$) and *unbinding* operations on those concepts via the vectors. One could bind "red" and "cat" to obtain a "red cat". The vectors can also be added, so "red" $\circledast$ "cat" + "yellow" $\circledast$ "dog" represents a "red cat and yellow dog". An inverse operator $\dagger$ is used to perform unbinding. One can then query a bound representation, asking "what was red?" by unbinding "red cat and yellow dog" $\circledast$ "red"† to get a vector $\approx$ "cat", where the resulting vector is necessarily corrupted by the noise by combining multiple vectors into a single fixed size representation.

To perform this symbolic manipulation the binding operation can be defined as $\mathcal{B} = \mathbf{x} \circledast \mathbf{y} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}_i) \odot \mathcal{F}(\mathbf{y}_i))$, where $\mathcal{F}$ denotes the FFT and $\odot$ an element-wise multiplication[1]. The inversion is defined as $\mathbf{y}^\dagger = \mathcal{F}^{-1}\left(\frac{1}{\mathcal{F}(\mathbf{y})}\right)$.

Combined Plate showed that the response $\mathcal{B}^\top \mathbf{y}^\dagger$ should be ≈ 1 if the vector $\mathbf{y} \in \mathcal{B}$, and ≈ 0 if not present. These properties hold in expectation provided that all vectors satisfy the sufficient condition that their elements are I.I.D. sampled from a Gaussian with zero mean and variance $1/H$, where H is the dimension of the vectors.

We will now show how to apply the same general logic of attention using HRR operations, creating an alternative (but not mathematically equivalent) form of self-attention that runs in linear time with respect to the sequence length. This is a slight "abuse" of the HRR, as our vectors will not be I.I.D. sampled random values, but results from prior layers in the network. Our design circumvents this issue in practice, which we will discuss shortly. We note this is a satisfying, but not required condition. Deviating from this adds more noise (our vectors are the outputs of prior layers in the network), but a softmax operation will act as a cleanup step to work without this condition.

Attention can be represented using queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$ matrices where the final output is computed as the weighted sum of the values. A query vector can be mapped to a set of linked key-value pairs to retrieve the value vector associated with the associated key. The concept of *binding* and *unbinding* operations of HRR is applied to link the key-value pair (i.e., bind the terms together), and then query a single representation of all key-value pairs to find the response values. For this reason, we will define the steps in an element-by-element manner that more naturally corresponds to the HRR operations, but our implementation will work in a batched manner. For this reason, we will discuss a single query $\boldsymbol{q}_t \in \mathbb{R}^H$, against the set of T key/value pairs $\boldsymbol{k}_t, \boldsymbol{v}_t \in \mathbb{R}^H$, where H is the dimension of the representation and $t \in 1, 2, \cdots T$. Thus $\boldsymbol{K} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots \boldsymbol{k}_T]$ is a matrix of shape (T, H), and similar for $\boldsymbol{Q}$ and $\boldsymbol{V}$.

First, we will create a superposition $\boldsymbol{\beta} \in \mathbb{R}^H$ of the key-value pairs, meaning that all vectors entering the superposition $\boldsymbol{\beta}$ are also similar (to some degree) to the final result. This is done by binding ($\circledast$) each key-value pair to associate them, and summing the results to form the superposition:

$$\boldsymbol{\beta} = \sum_{i=1}^{T} \boldsymbol{k}_i \circledast \boldsymbol{v}_i \tag{1}$$

$\boldsymbol{\beta}$ lets us compute interaction effects against all key-value pairs in one $\mathcal{O}(TH \log H)$ operation, avoiding the $\mathcal{O}(T^2 H)$ cost of explicit cross-correlation.

This now gives us a single vector $\boldsymbol{\beta}$ that represents the entire

[1] This is faster than an equivalent reformulation as multiplication by a circulant matrix of only real values.

sequence of T different key-value pair bindings. Now for each query we are interested in, we can obtain a vector that approximately matches the values $\boldsymbol{v}_{1,2,\ldots,T}$ via the symbolic property of HRRs that $\boldsymbol{x}^\dagger \circledast (\boldsymbol{x} \circledast \boldsymbol{y} + \boldsymbol{a} \circledast \boldsymbol{b}) \approx \boldsymbol{y}$, giving:

$$\hat{\boldsymbol{v}}_t = \boldsymbol{q}_t{}^\dagger \circledast \boldsymbol{\beta} \tag{2}$$

The queries are checked against the representation of all key-value pairs $\boldsymbol{\beta}$, where each $\boldsymbol{q}_t$ will contribute a corresponding value based on the response of the bound key, and the HRR framework allows us to perform them jointly. This now gives us a representation $\hat{\boldsymbol{v}}_t \in \mathbb{R}^H$ that represents the set of values present given the keys that respond to the input queries. We can then approximately determine the values present using the dot-product test that present values should result in ≈ 1 scalars, performing:

$$a_t = \text{cosine-similarity}\left(\boldsymbol{v}_t, \hat{\boldsymbol{v}}_t\right) \tag{3}$$

Each a_t is a scalar given the match between the original value $\boldsymbol{v}_t$ against the HRR extracted $\hat{\boldsymbol{v}}_t$, and is repeated for all T values to give us a response on the relative magnitude of each value present. With these approximate responses, we can compute a weighted distribution $\boldsymbol{w} \in \mathbb{R}^T$ by computing the softmax over all $a_{1,2,\ldots,T}$ responses, giving $\boldsymbol{w} = \text{softmax}(a_1, a_2, \ldots, a_T)$[2]. While each a_t will be highly noisy due to the inherent noise of HRR's superposition $\boldsymbol{\beta}$, and an amplified level of noise due to the use of non-I.I.D. Gaussian elements, the softmax has the practical effect of removing this noise for us. This occurs because the HRR results in similar magnitude noise across each a_t, and the softmax operation is invariant to constant additions to all elements.

For notational convenience to express this in more detail, let $\tilde{\Pi}_h(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ denote the pairwise interactions of the h'th term in evaluating an expression of the form $\left(\sum_{i=1}^{T} \boldsymbol{x}_i \circledast \boldsymbol{x}_{i+T}\right)^\top \boldsymbol{q}^T$, where all bold symbols are H dimensional vectors. The response of any query of the form $\boldsymbol{q} = \boldsymbol{x}_m + \boldsymbol{z}$ takes the form $\frac{\sum_{h=1}^{H}(x_{m,h}+z_h)\tilde{\Pi}_h(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k)(-1)^{h+1}}{\left(\sum_{h=1}^{H}(-1)^{h+1}x_{m,h}+\sum_{h=1}^{H}(-1)^{h+1}z_h\right)\left(\sum_{h=1}^{H}x_{m,h}+z_h\right)}$. In doing so we see that any noise vector $\boldsymbol{z}$ has a similar magnitude impact regardless of the target vector $\boldsymbol{x}_m$. Because the softmax is invariant to uniform magnitude adjustments to all inputs, and we have the same noise occurring for each computation, we get the behavior of the softmax effectively denoising the response due to the magnitude impacts. We discuss this further in Appendix D.

This softmax-based cleanup step is necessary because attempting to use $\hat{\boldsymbol{v}}_t$ directly results in degenerate random-guessing performance due to the noise of the HRR steps.

[2]We find no meaningful difference in results when using a temperature $\text{softmax}(\exp(\alpha)[a_1, \ldots, a_T])$.

With $\boldsymbol{w}$ in hand, we obtain the final Attention result

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [w_1\boldsymbol{v}_1, w_2\boldsymbol{v}_2, \ldots, w_T\boldsymbol{v}_T,] \tag{4}$$

returning a weighted version of the original values $\boldsymbol{V}$, approximating the standard attention's response. Critically, this process is linear in T and approximates an all pairs interaction between queries and keys, as shown by Theorem A.1

The rest of self-attention works in the same manner as the standard Transformer. The Attention function's inputs and outputs are altered by linear layers, and instead of performing single attention, we split the feature vector H of the query, key, and value into h heads each having a feature size of $H' = H/h$. The attention is computed in parallel in each head and then merged into single attention which is projected to get the final output. The Hrrformer is implemented using JAX and a code snippet of the self-attention mechanism is presented in Appendix A. The block diagram representation of the Hrrformer self-attention is presented in Figure 2. The diagram is shown for single head and single batch elements for brevity. A high-level overview of the architecture in a multi-head setting is presented in Figure 3 showing the analogy between Hrrformer and Transformer.

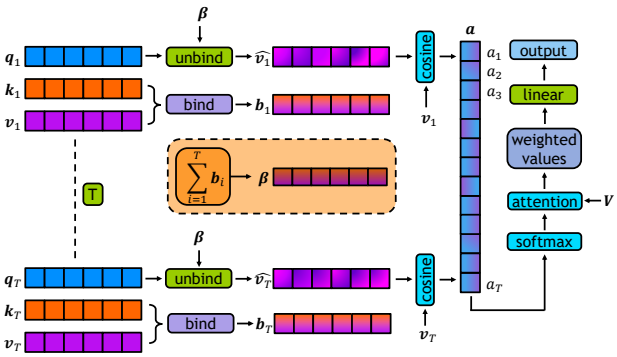

Figure 2: The block diagram of the Hrrformer self-attention. The *dashed* straight line represents the continuation of the same process for each T element. After computing the cosine similarity score vector $\mathbf{a}$, softmax is applied to compute the final attention weights $\mathbf{w}$ which is elementwise multiplied with value matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots \boldsymbol{v}_T]$. Afterward, a linear layer is used to get the final output.

The time complexity of the binding/unbinding operation is $\mathcal{O}(H \log H)$, which is performed T times as the dominant cost. Therefore, the time and space complexity of the Hrrformer attention per layer is linear in sequence length T where the time complexity is $\mathcal{O}(TH \log H)$ and the space complexity is $\mathcal{O}(TH)$.

This simple approach allows us to have fully replicated the same overall logical goals and construction of the attention mechanism first proposed by (Vaswani et al., 2017). The

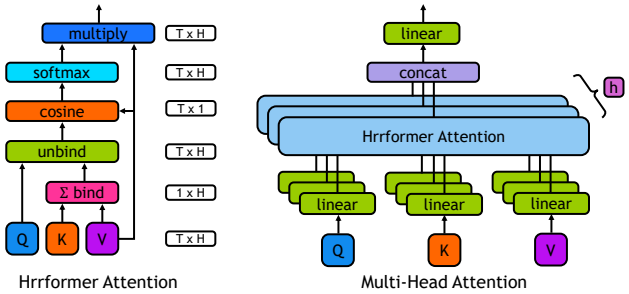

Figure 3: A high-level overview of our architecture, showing how the Hrrformer is analogous to the traditional transformer. Dataflow in a single-head with the shape of the tensor in different stages is shown on the left and multi-head attention is shown in right.

correspondence is not exact (e.g., returning weight original values instead of approximate value constructions), but allows us to avoid the non-I.I.D. issue of using arbitrary $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ as learned by the network. This neuro-symbolic reconstruction yields several benefits, as we will demonstrate in the next section. Simply replacing the self-attention in a standard Transformer with our HRR-based self-attention gives the "Hrrformer" that we will use to judge the utility of this new derivation.

4. Experiments and Results

The proposed Hrrformer is designed as an inexpensive alternative to the self-attention models for longer sequences. Experiments are performed to validate the effectiveness of the method in terms of time and space complexity in known benchmarks.

Our first result is running many of the current popular and state-of-the-art (SOTA) xformers on the real-world classification task of the Ember malware detection dataset (Anderson & Roth, 2018). This provides an example where the need to handle ever longer sequences exists and demonstrates that Hrrformer is one of the fastest and most accurate options on a problem with complex real-world dynamics. In doing so we also show that current SOTA methods such as Luna-256 do not generalize as well to new problem spaces, as our Hrrformer does.

Our second result will use the Long Range Arena (LRA) (Tay et al., 2020c) which has become a standard for evaluations in this space. The primary value of these results is to compare our Hrrformer with numerous prior works, establishing the broad benefits of faster time per epoch, convergence in $10\times$ fewer epochs, requiring only a single layer, and competitive overall accuracy. In addition, the LRA results are more accessible to the broader ML comunity and allow us to show visual evidence of HRR based attention learning to recover complex structure from a one-dimensional sequence.

4.1. EMBER

EMBER is a benchmark dataset for the malware classification task (Anderson & Roth, 2018). The benchmark contains $600K$ labeled training samples ($300K$ malicious, $300K$ benign) and $200K$ labeled test samples ($100K$ malicious, $100K$ benign). The maximum sequence length of this dataset is over $100M$ which is not feasible for any of the self-attention models to train with. We experiment with relatively shorter sequence lengths starting from $T = 256$ and doubling up to $T = 131072$ by truncating or padding the bytes until this maximum length is reached.

In this benchmark, Hrrformer is compared with Transformer (Vaswani et al., 2017), H-Transformer-1D (Zhu & Soricut, 2021), Luna-256 (Ma et al., 2021), Performer (Choromanski et al., 2020), Linformer (Wang et al., 2020), and F-Net (Lee-Thorp et al., 2021). All use 8 heads of a single encoder with 256 embedding size and 512 hidden size of the feed-forward network. Because this is a binary classification task, the encoder output is mapped into 2 logits output using back-to-back dense layers with ReLU activation. During training, the softmax cross-entropy loss function is optimized.

For sequence length 256, the batch size is set to be 256. In the experiment, as the sequence length doubles, we halved the batch size to fit the data and the model to the memory which can be expressed as $\max(2^{16-\log_2 T}, 1)$. This is done to push other models to the maximum possible length, and keep the batch size consistent between experiments. Additionally, a timeout limit of $10,000$s per epoch is set before experiments are terminated. The dropout rate is chosen to be 0.1, the learning rate is 10^{-3} with an exponential decay rate of 0.85. Each of the models is trained for a total of 10 epochs in 16 NVIDIA TESLA PH402 32GB GPUs.

Figure 1 shows the classification accuracy of each of the methods for incremental sequence length from 512 to 131072. As the sequence length increases, Hrrformer outperforms the rest of the models achieving the highest 91.03% accuracy for maximum sequence length 16384. In terms of execution time F-Net is the only model that is faster than ours, however the accuracy of F-Net is an absolute 4.53% points lower (Table 1). Even after exponentially decaying batch size, we could not fit the standard Transformer model to the memory for the sequence length 8196 indicating out-of-memory (OOM) in all figures. H-transformer-1d and Luna-256 crossed the timeout limit for sequence length 16384 indicated out-of-time (OOT) in the figure. The detailed numeric results are presented in Appendix B with additional results for the sequence length of 256. The execution time for linear time complexity methods seems quadratic in

the figure; this is due to the exponential decay of the batch size with the increase of sequence length, which was necessary to push each model to its maximum possible sequence length. The more detailed timing information can be seen in Figure 4, where all models but F-Net and Hrrformer run out of time or memory before reaching the maximum sequence length. Note as well that as the sequence length increases, the already small difference in runtime between F-Net and Hrrformer reduces to near-zero.

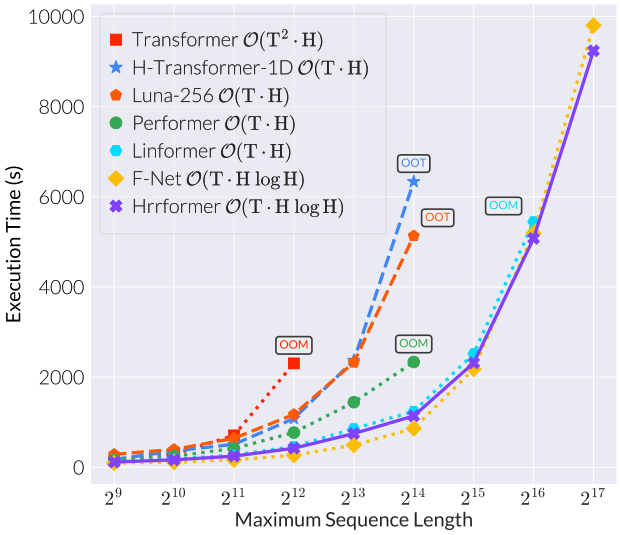

Figure 4: The total runtime on the Ember dataset for each algorithm, with the big-$\mathcal{O}$ runtime complexity associated. While Hrrformer is technically a slower big-$\mathcal{O}$ due to the extra $\log H$ term, the hidden size of the network is generally fixed and smaller than the sequence length. Thus we see in practice our design allows for faster execution in training and inference. Most prior methods fail early by running Out Of Memory (OOM) or Time (OOT).

Of significant importance to our results is that Luna-256 performs considerably worse than all other options, compared to its top accuracy in the LRA. We hypothesize that the Ember task requires more complex reasoning and feature extraction over time and because Luna performs aggressive compression and approximation of the time component of the model it suffers in terms of accuracy. Our Hrrformer on the other hand has consistent behavior across Ember and the LRA: high accuracy, able to handle longer sequences, and convergence in few epochs, a requirement for working on this dataset which is 1 TB in size and is otherwise prohibitive in its scale.

4.2. Long Range Arena

The Long Range Arena (LRA) (Tay et al., 2020c) benchmark comprises 6 diverse tasks covering image, text, math, language, and spatial modeling under long context scenarios ranging from $1K$ to $16K$. **ListOps** – task inspects the capability of modeling hierarchically structured data in a longer sequence context with mathematical operators `MAX`, `MEAN`, `MEDIAN`, and `SUM MOD` enclosed by delimiters. This is a ten-way classification problem with a maximum sequence length of $2K$. **Text** – is a byte/character level classification task using the IMDB movie review (Maas et al., 2011) dataset. Character-level language modeling makes the models reason with compositional unsegmented data.This is a binary classification task with a maximum sequence length of $4K$. **Retrieval** – evaluates the model's ability to encode and compress useful information for matching and retrieval by modeling similarity score between two documents. For this task, the ACL Anthology Network (Radev et al., 2013) dataset is used in a character level setup. This task has a maximum sequence length of $8K$ and this is a binary classification task. **Image** – is an image classification task of 10 classes that uses grayscale CIFAR-10 dataset in a sequence of length $32 \times 32 = 1024$. This task allows assessing the model's ability to process discrete symbols. **Pathfinder** – task evaluates the model's performance over long-range spatial dependency. This is a binary classification task that classifies whether two circles are connected by a line which is introduced in (Linsley et al., 2018), and includes distractor paths. The images have dimension 32×32 which is reshaped into 1024. **Path-X** - is extremely difficult version of pathfinder task which contains images of dimension $128 \times 128 = 16384$ with additional distractor paths.

In Hrrformer, we use the same number or fewer parameters as mentioned in the LRA benchmark (Tay et al., 2020c) across the tasks and a list of hyper-parameters used in each task is provided in Appendix B. Global average pooling is applied to the output of the encoder sequences and subsequently back to back dense layers are used with ReLU activation to get the final logits output. During training, the softmax cross-entropy loss function is optimized using the Adam optimizer. We use the exponential decay learning rate with the initial value of 10^{-3}, and the final value of 10^{-5}. For all the tasks, Hrrformer is trained for a total of 20 epochs both in the case of single- and multi-layer which is $10\times$ less training than previous works. The results in terms of accuracy in all the tasks of the LRA benchmark are presented in Table 1. [3]

Ours is one of only two methods that improve accuracy upon the Transformer and consistently displayed higher per-

[3]The Pathfinder task as originally reported by (Tay et al., 2020c) uses a "hard" version of the task, but the code provided defaults to an "easy" version. Most papers do not make clear which version of the task is evaluated, and the F-Net authors indicated in correspondence the "easy" version was used. Luna-256 used the hard version, and other authors have not yet reached back to us. On the easy version, Hrrformer gets 80.81% in a single-layer and 80.77% in the multi-layer, but we report the hard version in our table and assume others are using the hard version.

Table 1: Accuracy results of Hrrformer on Long Range Arena (LRA) benchmark. Even using just one layer Hrrformer is highly competitive, and the only method besides Luna is a Pareto improvement over the original Transformer. Our method is further advantaged in that it requires $10\times$ fewer epochs to reach competitive accuracies. Best results in **bold**, second best in *italics*.

Model	ListOps (2k)	Text (4k)	Retrieval (4k)	Image (1k)	Path (1k)	Path-X (16k)	Avg	Epochs
Transformer (Vaswani et al., 2017)	36.37	64.27	57.46	42.44	71.40	FAIL	54.39	200
Local Attention (Tay et al., 2020c)	15.82	52.98	53.39	41.46	66.63	FAIL	46.06	200
Linear Transformer (Katharopoulos et al., 2020)	16.13	65.90	53.09	42.34	75.30	FAIL	50.55	200
Reformer (Kitaev et al., 2020)	37.27	56.10	53.40	38.07	68.50	FAIL	50.67	200
Sparse Transformer (Child et al., 2019)	17.07	63.58	59.59	44.24	71.71	FAIL	51.24	200
Sinkhorn Transformer (Tay et al., 2020b)	33.67	61.20	53.83	41.23	67.45	FAIL	51.29	200
Linformer (Wang et al., 2020)	35.70	53.94	52.27	38.56	76.34	FAIL	51.36	200
Performer (Choromanski et al., 2020)	18.01	65.40	53.82	42.77	77.05	FAIL	51.41	200
Synthesizer (Tay et al., 2020a)	36.99	61.68	54.67	41.61	69.45	FAIL	52.88	200
Longformer (Beltagy et al., 2020)	35.63	62.85	56.89	42.22	69.71	FAIL	53.46	200
BigBird (Zaheer et al., 2020)	36.05	64.02	59.29	40.83	74.87	FAIL	55.01	200
F-Net (Lee-Thorp et al., 2021)	35.33	65.11	59.61	38.67	*77.78*	FAIL	54.42	200
Nystromformer (Xiong et al., 2021)	37.15	65.52	**79.56**	41.58	70.94	FAIL	58.95	200
Luna-256 (Ma et al., 2021)	37.98	65.78	79.56	47.86	**78.55**	FAIL	**61.95**	200
H-Transformer-1D (Zhu & Soricut, 2021)	**49.53**	**78.69**	63.99	46.05	68.78	FAIL	61.41	200
Hrrformer Single-layer	38.79	66.50	75.40	48.47	70.71	FAIL	59.97	**20**
Hrrformer Multi-layer	39.98	65.38	76.15	**50.45**	72.17	FAIL	60.83	**20**

formance in all the tasks. We show the performance for both single and multiple layers. In 3 of the 5 tasks (ListOps, Text, Image), Hrrformer achieves the second-best results using only 1 layer of the encoder. For the Image classification task, it achieves the best results of 50.45% accuracy using 3 layers of the encoder. Moreover, Hrrformer requires $10\times$ fewer epochs than others to produce comparable or better results. Overall, the multi-layered Hrrformer produces the third-best result of 60.83% in the benchmark.

The ability to learn with a single layer aids in both throughput and memory use. The result is surprising, and in visualizing the weight vector $\mathbf{w}$ we can confirm that a single layer is sufficient to learn the structure. We show this for the Image task of single-layer Hrrformer in Figure 5 (multi-layer in Appendix C). Here, the weight vector $\mathbf{w} \in \mathbb{R}^{1024\times 1}$ is reshaped to 32×32, the shape of the original grayscale images of the CIFAR-10 dataset for visualization. From the figure, it is clear that the Hrrformer is learning to identify the 2D structure from the 1D sequence of the Image classification task. We also compare against the standard Transformer in Appendix Figure 10, where it is less obvious how the model's weights might correspond to the 2D structure of the image.

Hrrformer's benefits go beyond accuracy and convergence speed: it is fast and consumes the least amount of memory on GPU of the alternatives tested. Figure 6 compares all the self-attention models in terms of LRA score, speed (training examples per second), and memory footprint (size of the circle). LRA score is the mean accuracy of all the tasks in the LRA benchmark. Speed and memory footprint is calculated on the byte-level text classification task per epoch. To measure these results, a single NVIDIA TESLA PH402 32GB GPU is utilized with a fixed batch size of 4 and a maximum sequence length of 4000 with an embedding size

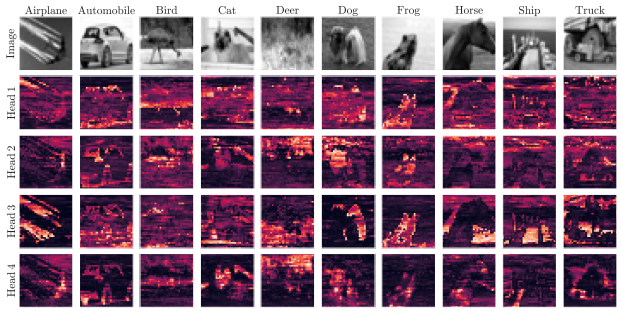

Figure 5: Visualization of weight vector $\mathbf{w} \in \mathbb{R}^{1024\times 1}$ reshaped to 32×32, the shape of the original image of the CIFAR-10 dataset used in the LRA Image classification task. A single-layer Hrrformer is able to learn the 2D structure from the 1D sequence of the image. This is particularly noticeable in the Airplane, dog, Frog, and Horse images. Note context sensitive Head activation can be observed comparing Head 3 for dog vs Frog, where activation occurs for different pixel intensities indicating the model is not naively activating for simple color intensity.

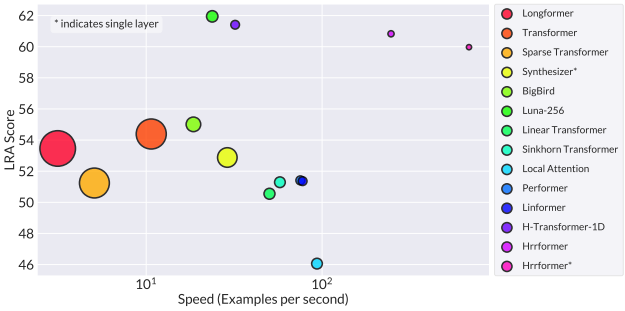

Figure 6: Performance (y-axis), Speed (x-axis, log-scale) of different xformers, and memory footprint on GPU are illustrated by the size of the circles. Hrrformer is in the top-right of the graph, with the smallest circle size, indicating it is the fastest and most memory efficient for training (this does *not* factor in convergence speed).

of 32 and feature size of 64. For all the models 6 layers of the encoder are used. Both single- and multi-layered Hrrformer are $28\times$ and $10\times$ faster than the Luna-256 (Ma et al., 2021) which has achieved the highest accuracy in the LRA benchmark. Hrrformer also consumes the least amount of memory, taking 79.15% and 70.66% less memory compared to Luna-256 in the case of single and multi-layered Hrrformer, respectively. The detailed numeric results of Figure 6 are given in Appendix B.

Hrrformer also reduces the amount of overfitting between training and test performance. We compare the training and test accuracy, and amount of overfitting of the Image classification task to the other self-attention models presented in LRA benchmark (Tay et al., 2020c) and for which data are available[4]. Table 2 exhibits that the Hrrformer acquires the best results on the test set with an 6.83% train/test gap. The learning curves of all the task is also presented in Appendix Figure 8 demonstrating the lower overfitting nature of the Hrrformer across the tasks.

Hrrformer's inference time is also faster than other options for long sequences. As an example, the time to make predictions for the text classification task is given in Appendix Table 7, where the single-layer Hrrformer is the fastest option, followed by the multi-layer Hrrformer. We also find Hrrformer's inference time is relatively faster regardless of the batch size. The inference time for the Hrrformer with a batch size of 2 is still $5\times$ faster than the inference time for the Transformer with a batch size of 32. More details are presented in Appendix Table 6.

[4]We do not have the compute resources to run the other xformers on the LRA ourselves, in part due to the higher memory use that exceeds our infrastructure.

Table 2: Training and test accuracy of different self-attention models on the Image classification task. Among all the models, Hrrformer achieves the best test accuracy with the least amount of overfitting (lower is better).

Model	Train Accuracy (%) ↑	Test Accuracy (%) ↑	Overfitting (%) ↓
Transformer	69.45	42.44	27.01
Local Attention	63.19	41.46	21.73
Sparse Transformer	66.74	44.24	22.50
Longformer	71.65	42.22	29.43
Linformer	97.23	38.56	58.67
Reformer	68.45	38.07	30.38
Sinkhorn Transformer	69.21	41.23	27.98
Synthesizer	**97.31**	41.61	55.70
BigBird	71.49	40.83	30.66
Linear Transformer	65.61	42.34	23.27
Performer	73.90	42.77	31.13
Hrrformer	57.28	**50.45**	**6.83**

5. Conclusion

The Hrrformer is a neuro-symbolic reconstruction of self-attention. The proposed method is faster in compute and consumes less memory per layer. We have tested Hrrformer on known LRA and EMBER benchmarks. In the LRA benchmark, Hrrformer has achieved the near state-of-the-art accuracy of 60.83% using a single layer of an encoder. In terms of speed, it is $28\times$ and $10\times$ faster than the current SOTA in the case of single and multiple layers, respectively. Additionally, it takes 79.15% and 70.66% less memory on GPU compared to Luna-256 for single and multiple layers of Hrrformer. Moreover, it converges $10\times$ faster than other self-attention models. In the EMBER malware classification dataset, Hrrformer attained the highest 91.03% accuracy for a maximum sequence length of 16384 with a significantly faster processing rate. In conclusion, Hrrformer is $\approx 280\times$ faster to train and a single layer of the encoder is sufficient to learn the structure of the input.

References

Abou-Assaleh, T., Cercone, N., Keselj, V., and Sweidan, R. N-gram-based detection of new malicious code. In *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, volume 2, pp. 41–42. IEEE, 2004. ISBN 0-7695-2209-2. doi: 10.1109/CMPSAC.2004.1342667. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1342667.

Aghakhani, H., Gritti, F., Mecca, F., Lindorfer, M., Ortolani, S., Balzarotti, D., Vigna, G., and Kruegel, C. When Malware is Packin' Heat; Limits of Machine Learning Classifiers Based on Static Analysis Features. In *Proceedings 2020 Network and Distributed System Security Symposium*, Reston, VA, 2020. Internet Society. ISBN

1-891562-61-4. doi: 10.14722/ndss.2020.24310. URL https://www.ndss-symposium.org/wp-content/uploads/2020/02/24310.pdf.

Anderson, H. S. and Roth, P. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T., Rasmussen, D., Choo, X., Voelker, A., and Eliasmith, C. Nengo: a Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, 7:48, 2014. ISSN 1662-5196. doi: 10.3389/fninf.2013.00048. URL https://www.frontiersin.org/article/10.3389/fninf.2013.00048.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Blouw, P. and Eliasmith, C. A Neurally Plausible Encoding of Word Order Information into a Semantic Vector Space. *35th Annual Conference of the Cognitive Science Society*, 35:1905–1910, 2013.

Blouw, P., Solodkin, E., Thagard, P., and Eliasmith, C. Concepts as Semantic Pointers: A Framework and Computational Model. *Cognitive Science*, 40(5):1128–1162, 7 2016. ISSN 03640213. doi: 10.1111/cogs.12265. URL http://doi.wiley.com/10.1111/cogs.12265.

Borbely, R. S. On normalized compression distance and large malware. *Journal of Computer Virology and Hacking Techniques*, pp. 1–8, 2015. ISSN 2263-8733. doi: 10.1007/s11416-015-0260-0.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., and Graves, A. Associative Long Short-Term Memory. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1986–1994, 2016.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. A Large-Scale Model of the Functioning Brain. *Science*, 338(6111):1202–1205, 11 2012. ISSN 0036-8075. doi: 10.1126/science.1225266. URL https://www.sciencemag.org/lookup/doi/10.1126/science.1225266.

Ganesan, A., Gao, H., Gandhi, S., Raff, E., Oates, T., Holt, J., and McLean, M. Learning with Holographic Reduced Representations. In *Advances in Neural Information Processing Systems*, 2021. URL http://arxiv.org/abs/2109.02157.

Goel, K., Gu, A., Donahue, C., and Ré, C. It's Raw! Audio Generation with State-Space Models. *arXiv*, pp. 1–23, 2022. URL http://arxiv.org/abs/2202.09729.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. HiPPO: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 2020. ISSN 10495258.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers. In *NeurIPS*, 2021. URL http://arxiv.org/abs/2110.13985.

Gu, A., Goel, K., and Ré, C. Efficiently Modeling Long Sequences with Structured State Spaces. In *ICLR*, 2022. URL http://arxiv.org/abs/2111.00396.

Huang, Q., Smolensky, P., He, X., Deng, L., and Wu, D. Tensor Product Generation Networks for Deep NLP Modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1263–1273, New Orleans, Louisiana, 6 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1114. URL https://aclanthology.org/N18-1114.

Jones, M. N. and Mewhort, D. J. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114(1):1–37, 2007. ISSN 0033295X. doi: 10.1037/0033-295X.114.1.1.

Kaban, A. Improved Bounds on the Dot Product under Random Projection and Random Sign Projection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 487–496, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783364. URL https://doi.org/10.1145/2783258.2783364.

Kantchelian, A., Tschantz, M. C., Huang, L., Bartlett, P. L., Joseph, A. D., and Tygar, J. D. Large-margin Convex Polytope Machine. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pp. 3248–3256, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969189.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Kephart, J. O., Sorkin, G. B., Arnold, W. C., Chess, D. M., Tesauro, G. J., and White, S. R. Biologically Inspired Defenses Against Computer Viruses. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pp. 985–996, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8, 978-1-558-60363-9. URL http://dl.acm.org/citation.cfm?id=1625855.1625983.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kolter, J. Z. and Maloof, M. A. Learning to Detect and Classify Malicious Executables in the Wild. *Journal of Machine Learning Research*, 7:2721–2744, 12 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248646.

Krčál, M., Švec, O., Bálek, M., and Jašek, O. Deep Convolutional Malware Classifiers Can Learn from Raw Executables and Labels Only. In *ICLR Workshop*, 2018.

Le, H., Tran, T., and Venkatesh, S. Self-Attentive Associative Memory. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5682–5691. PMLR, 2020. URL https://proceedings.mlr.press/v119/le20b.html.

Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontanon, S. FNet: Mixing Tokens with Fourier Transforms. *arXiv*, 2021. URL http://arxiv.org/abs/2105.03824.

Li, M., Chen, X., Li, X., Ma, B., and Vitanyi, P. M. The Similarity Metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004. ISSN 0018-9448. doi: 10.1109/TIT.2004.838101.

Linsley, D., Kim, J., Veerabadran, V., Windolf, C., and Serre, T. Learning long-range spatial dependencies with horizontal gated recurrent units. *Advances in neural information processing systems*, 31, 2018.

Liu, L. and Wang, B. Malware classification using gray-scale images and ensemble learning. In *2016 3rd International Conference on Systems and Informatics (ICSAI)*, pp. 1018–1022. IEEE, 11 2016. ISBN 978-1-5090-5521-0. doi: 10.1109/ICSAI.2016.7811100. URL http://ieeexplore.ieee.org/document/7811100/.

Ma, X., Kong, X., Wang, S., Zhou, C., May, J., Ma, H., and Zettlemoyer, L. Luna: Linear Unified Nested Attention. In *NeurIPS*, 2021. URL http://arxiv.org/abs/2106.01540.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Menéndez, H. D., Bhattacharya, S., Clark, D., and Barr, E. T. The arms race: Adversarial search defeats entropy used to detect malware. *Expert Systems with Applications*, 118:246–260, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.10.011. URL http://www.sciencedirect.com/science/article/pii/S0957417418306535.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. Malware Images: Visualization and Automatic Classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, pp. 4:1–4:7, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0679-9. doi: 10.1145/2016904.2016908. URL http://doi.acm.org/10.1145/2016904.2016908.

Nickel, M., Rosasco, L., and Poggio, T. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 1955–1961. AAAI Press, 2016.

Plate, T. A. Holographic Recurrent Networks. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'92, pp. 34–41, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 1558602747.

Radev, D. R., Muthukrishnan, P., Qazvinian, V., and Abu-Jbara, A. The acl anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, 2013.

Raff, E. and Nicholas, C. An Alternative to NCD for Large Sequences, Lempel-Ziv Jaccard Distance. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, pp. 1007–1015, New York, New York, USA, 2017. ACM Press. ISBN 9781450348874. doi: 10.1145/3097983.3098111. URL http://dl.acm.org/citation.cfm?doid=3097983.3098111.

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., and Nicholas, C. Malware Detection by Eating a Whole EXE. In *AAAI Workshop on Artificial Intelligence for Cyber Security*, 10 2018. URL http://arxiv.org/abs/1710.09435.

Raff, E., Fleming, W., Zak, R., Anderson, H., Finlayson, B., Nicholas, C. K., Mclean, M., Fleming, W., Nicholas, C. K., Zak, R., and Mclean, M. KiloGrams: Very Large N-Grams for Malware Classification. In *Proceedings of KDD 2019 Workshop on Learning and Mining for Cybersecurity (LEMINCS'19)*, 2019. URL https://arxiv.org/abs/1908.00200.

Raff, E., Nicholas, C., and McLean, M. A New Burrows Wheeler Transform Markov Distance. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 5444–5453, 2020. doi: 10.1609/aaai.v34i04.5994. URL http://arxiv.org/abs/1912.13046.

Raff, E., Fleshman, W., Zak, R., Anderson, H. S., Filar, B., and McLean, M. Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021. URL http://arxiv.org/abs/2012.09390.

Rahimi, A. and Recht, B. Random Features for Large-Scale Kernel Machines. In *Neural Information Processing Systems*, number 1, 2007. URL http://seattle.intel-research.net/pubs/rahimi-recht-random-features.pdf.

Rudd, E. M. and Abdallah, A. Training Transformers for Information Security Tasks: A Case Study on Malicious URL Prediction. *arXiv*, 2020. doi: 10.48550/arXiv.2011.03040. URL http://arxiv.org/abs/2011.03040.

Rudd, E. M., Rahman, M. S., and Tully, P. Transformers for End-to-End InfoSec Tasks: A Feasibility Study. In *Proceedings of the 1st Workshop on Robust Malware Analysis*, WoRMA '22, pp. 21–31, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391795. doi: 10.1145/3494110.3528242. URL https://doi.org/10.1145/3494110.3528242.

S. Resende, J., Martins, R., and Antunes, L. A Survey on Using Kolmogorov Complexity in Cybersecurity. *Entropy*, 21(12):1196, 12 2019. ISSN 1099-4300. doi: 10.3390/e21121196. URL https://www.mdpi.com/1099-4300/21/12/1196.

Schlag, I. and Schmidhuber, J. Learning to Reason with Third Order Tensor Products. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/a274315e1abede44d63005826249d1df-Paper.pdf.

Si, S., Hsieh, C.-J., and Dhillon, I. S. Computationally Efficient Nystrom Approximation using Fast Transforms. In *International Conference on Machine Learning (ICML)*, 6 2016.

Singh, R. and Eliasmith, C. Higher-Dimensional Neurons Explain the Tuning and Dynamics of Working Memory Cells. *Journal of Neuroscience*, 26(14):3667–3678, 2006. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.4864-05.2006. URL https://www.jneurosci.org/content/26/14/3667.

Sinha, A. and Duchi, J. C. Learning Kernels with Random Features. In Lee, D. D., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances In Neural Information Processing Systems 29*, pp. 1298–1306. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6180-learning-kernels-with-random-features.pdf.

Smolensky, P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1):159–216, 1990. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(90)90007-M. URL https://www.sciencedirect.com/science/article/pii/0004370290900 07M.

Sourkov, V. IGLOO: Slicing the Features Space to Represent Long Sequences. *arXiv*, 2018. URL http://arxiv.org/abs/1807.03402.

Stewart, T. C. and Eliasmith, C. Large-scale synthesis of functional spiking neural circuits. *Proceedings of the IEEE*, 102(5):881–898, 2014. ISSN 00189219. doi: 10.1109/JPROC.2014.2306061.

Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. Adaptive Attention Span in Transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 331–335, Florence, Italy, 7 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1032. URL https://aclanthology.org/P19-1032.

Tay, Y., Bahri, D., Metzler, D., Juan, D., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2020a.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pp. 9438–9447. PMLR, 2020b.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020c.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Voelker, A., Kajić, I., and Eliasmith, C. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d\textquotesingle Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 15544–15553. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9689-legendre-memory-units-continuous-time-representation-in-recurrent-neural-networks.pdf.

Walenstein, A. and Lakhotia, A. The Software Similarity Problem in Malware Analysis. *Duplication, Redundancy, and Similarity in Software*, 2007. URL http://drops.dagstuhl.de/opus/volltexte/2007/964.

Wang, J., Wonka, P., and Ye, J. Scaling SVM and Least Absolute Deviations via Exact Data Reduction. *JMLR W&CP*, 32(1):523–531, 2014.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, Z., Crammer, K., and Vucetic, S. Multi-class pegasos on a budget. In *27th International Conference on Machine Learning*, pp. 1143–1150, 2010. URL http://www.ist.temple.edu/~vucetic/documents/wang10icml.pdf.

Wang, Z., Djuric, N., Crammer, K., and Vucetic, S. Trading representability for scalability Adaptive Multi-Hyperplane Machine for nonlinear Classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, pp. 24, New York, New York, USA, 2011. ACM Press. ISBN 9781450308137. doi: 10.1145/2020408.2020420. URL http://dl.acm.org/citation.cfm?doid=2020408.2020420.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nystr\"omformer: A Nystr\"om-Based Algorithm for Approximating Self-Attention. In *AAAI*, 2021. URL http://arxiv.org/abs/2102.03902.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Zak, R., Raff, E., and Nicholas, C. What can N-grams learn for malware detection? In *2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 109–118. IEEE, 10 2017. ISBN 978-1-5386-1436-5. doi: 10.1109/MALWARE.2017.8323963. URL http://ieeexplore.ieee.org/document/8323963/.

Zhu, Z. and Soricut, R. H-Transformer-1D: Fast One-Dimensional Hierarchical Attention for Sequences. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3801–3815, Online, 8 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.294. URL https://aclanthology.org/2021.acl-long.294.

A. Self-attention Definition

The code of the Hrrformer self-attention model is written in JAX. Below is a code snippet of the Multi-headed Hrrformer attention. The shape of the output vector of each line is given by a comment where B, T, and H represent the batch size, maximum sequence length, and feature size, respectively. h is the number of heads and H' is the feature dimension in each head.

```
class SelfAttention(nn.Module):
    features: int
    heads: int

    def setup(self):
        self.binding = Binding()
        self.unbinding = Unbinding()
        self.similarity = CosineSimilarity()

    @nn.compact
    def __call__(self, inputs, mask=None):
        dense = partial(nn.DenseGeneral, features=self.features, use_bias=False)

        q, k, v = (dense(name='query')(inputs),   # (B, T, H)
                   dense(name='key')(inputs),     # (B, T, H)
                   dense(name='value')(inputs))   # (B, T, H)

        q, k, v = (split(q, self.heads),   # (B, h, T, H')
                   split(k, self.heads),   # (B, h, T, H')
                   split(v, self.heads))   # (B, h, T, H')

        bind = self.binding(k, v, axis=-1)            # (B, h, T, H')
        bind = np.sum(bind, axis=-2, keepdims=True)   # (B, h, 1, H')

        vp = self.unbinding(bind, q, axis=-1)                    # (B, h, T, H')
        scale = self.similarity(v, vp, axis=-1, keepdims=True)   # (B, h, T, 1)

        if mask is not None:
            scale = scale + (1. - mask) * (-1e9)   # (B, h, T, 1)
        weight = nn.softmax(scale, axis=-2)        # (B, h, T, 1)
        weighted_value = weight * v                # (B, h, 1, H')

        weighted_value = merge(weighted_value)          # (B, T, H)
        output = dense(name='output')(weighted_value)   # (B, T, H)
        return output
```

Figure 7: Multi-headed Hrrformer Self-attention.

Theorem A.1. *The Hrrformer Attention approximates an all-pairs interaction between all queries and key-values.*

Proof. Expand Equation 3 as cosine-sim $\left(\boldsymbol{v}_t, \boldsymbol{q}_t{}^{\dagger} \circledast \left(\sum_{i=1}^{T} \boldsymbol{k}_i \circledast \boldsymbol{v}_i\right)\right)$. The distributive property of the binding operation $\circledast$ allows us to move the query inside summation, producing cosine-sim $\left(\boldsymbol{v}_t, \sum_{i=1}^{T} \boldsymbol{q}_i^{\dagger} \circledast \boldsymbol{k}_i \circledast \boldsymbol{v}_i\right)$. At the cost of noise terms not specified, we can see that the response of the cosine similarity is produced from an interaction between the time step t and summation of all query-key pairs for $1, 2, \cdots, T$ steps, showing that a cross-product is approximated by the Hrrformer. □

B. Hyperparameters & Numeric Results

The hyperparameters used in each task of the Long Range Arena (LRA) benchmark and EMBER malware classification task are presented in Table 3. In all of the tasks, the Adam optimizer is used with an exponential decay learning rate. The starting learning rate is 10^{-3} and the final learning rate is 10^{-5}. The decay rate indicates the amount of learning rate decay per epoch. MLP dim indicates the number of features used in the first linear layer of the MLP block after the attention block.

Table 3: List of the hyperparameters used in the Long Range Arena (LRA) benchmark and EMBER malware classification task.

Task	Positional Embedding	Batch size	Vocab size	Maximum Sequence Length	Embed dim	MLP dim	Heads	Layers	Classes	Decay rate
ListOps	Learned	32	17	2000	512	256	8	6	10	0.90
Text	Fixed	32	257	4000	512	1024	8	6	2	0.90
Retrieval	Fixed	64	257	4000	128	64	4	4	2	0.90
Image	Fixed	32	256	1024	256	128	4	3	10	0.95
Path	Learned	128	256	1024	1024	256	8	2	2	0.95
Malware	Learned	$\max(2^{16-\log_2 T},\ 1)$	257	T	256	512	8	1	2	0.85

The detailed results of the Hrrformer of Figures 6 are presented here. The numerical results of the comparison of Hrrformer with other self-attention models in terms of LRA score, speed (examples per second), and memory footprint (size of the circle) are presented in Table 4. From the table, it can be observed that the Hrrformer only lags 1.12% behind Luna-256 (Ma et al., 2021) in the LRA score. However, in terms of speed, single- and multi-layered Hrrformer are $28\times$ and $10\times$ faster than Luna-256. Moreover, Hrrformer consumes 79.15% and 70.66% less memory than Luna-256 in the case of single and multi-layered Hrrformer, respectively. The numerical results of EMBER malware classification are presented in Table 5. From the table, it can be observed that as the sequence length increases, Hrrformer surpasses the other models, and for the sequence length $16,384$, has achieved the highest accuracy of 91.03%.

Table 4: LRA score, speed in examples per second, and total memory usage in MB of all the different xformer models used in LRA benchmark. The speed and memory usage metrics are computed using 6 layers of encoder in byte-level text classification task. In the chart, * indicates the use of single layer of encoder. Best results in **bold**, second best in **italics**.

Model	LRA Score Accuracy (%)	Speed (Examples per Second)	Time (s)	Memory Usage (MB)
Longformer	53.46	3.14	7959.42	30978.21
Sparse Transformer	51.24	5.08	4923.98	21876.57
Transformer	54.39	10.68	2340.31	22134.52
BigBird	55.01	18.56	1347.26	5218.89
Luna-256	**61.95**	23.74	1053.25	3184.66
Synthesizer*	52.88	28.92	864.45	9377.55
H-Transformer-1D	<u>61.41</u>	32.03	780.42	1838.28
Linear Transformer	50.55	50.22	497.84	2941.39
Sinkhorn Transformer	51.29	57.56	434.31	2800.88
Performer	51.41	75.23	332.31	1951.53
Linformer	51.36	77.49	322.62	1867.64
Local Attention	46.06	93.51	267.35	2800.88
Hrrformer	60.83	<u>246.45</u>	<u>101.44</u>	<u>934.41</u>
Hrrformer*	59.97	**683.81**	**36.56**	**663.88**

In addition we provide the time to perform inference over the entire LRA text classification task for batch sizes varying between 2 and 32. This is shown in Table 6, where the time decreases as batch size increases due to reduced overhead and higher GPU compute efficiency. As can be seen the Hrrformer is uniformly faster, and more consistent in total run-time. Similarly, our method is faster for larger and small batch sizes, a particularly valuable benefit in inference where batching is not always possible. This can be seen in Table 7, where the inference time for the Hrrformer with a batch size of 2 is still $5\times$ faster than the inference time for the Transformer with a batch size of 32.

Table 5: Accuracy and the execution time of the different self-attention models for different sequence lengths in the EMBER malware classification dataset. Best results in **bold**.

Model		Maximum Sequence Length									
		256	512	1,024	2,048	4,096	8,192	16,384	32,768	65,536	131,072
Transformer	Accuracy (%)	74.87	84.27	86.74	87.76	88.43	–	–	–	–	–
	Time (s)	101.59	146.96	286.98	708.7	2305.28	–	–	–	–	–
H-Transformer-1D	Accuracy (%)	59.59	78.17	85.45	87.8	90.14	88.9	90.48	–	–	–
	Time (s)	116.6	175.04	362.41	509.63	1082.67	2371.96	6336.37	–	–	–
Luna-256	Accuracy (%)	70.21	74.8	77.01	80.06	79.18	83.76	83.55	–	–	–
	Time (s)	243.04	287.5	395.87	643.81	1172.35	2326.15	5132.95	–	–	–
Performer	Accuracy (%)	78.0	**87.74**	**88.91**	**89.77**	89.06	89.88	85.68	–	–	–
	Time (s)	115.77	159.59	247.02	418.1	770.75	1444.38	2334.94	–	–	–
Linformer	Accuracy (%)	**79.52**	86.41	88.73	88.25	86.57	86.53	86.94	85.70	83.75	–
	Time (s)	99.18	124.66	179.56	273.71	459.68	855.85	1239.88	2518.44	5445.57	–
F-Net	Accuracy (%)	76.42	80.25	80.87	84.34	83.55	86.36	86.00	86.29	86.45	86.40
	Time (s)	84.84	95.58	113.2	165.77	267.21	492.44	861.48	2182.30	5191.26	9800.97
Hrrformer	Accuracy (%)	78.06	83.95	88.07	89.22	**90.59**	**90.89**	**91.03**	**90.65**	**90.13**	**89.46**
	Time (s)	91.35	117.96	165.18	247.32	423.55	748.48	1138.75	2315.62	5076.65	9237.78

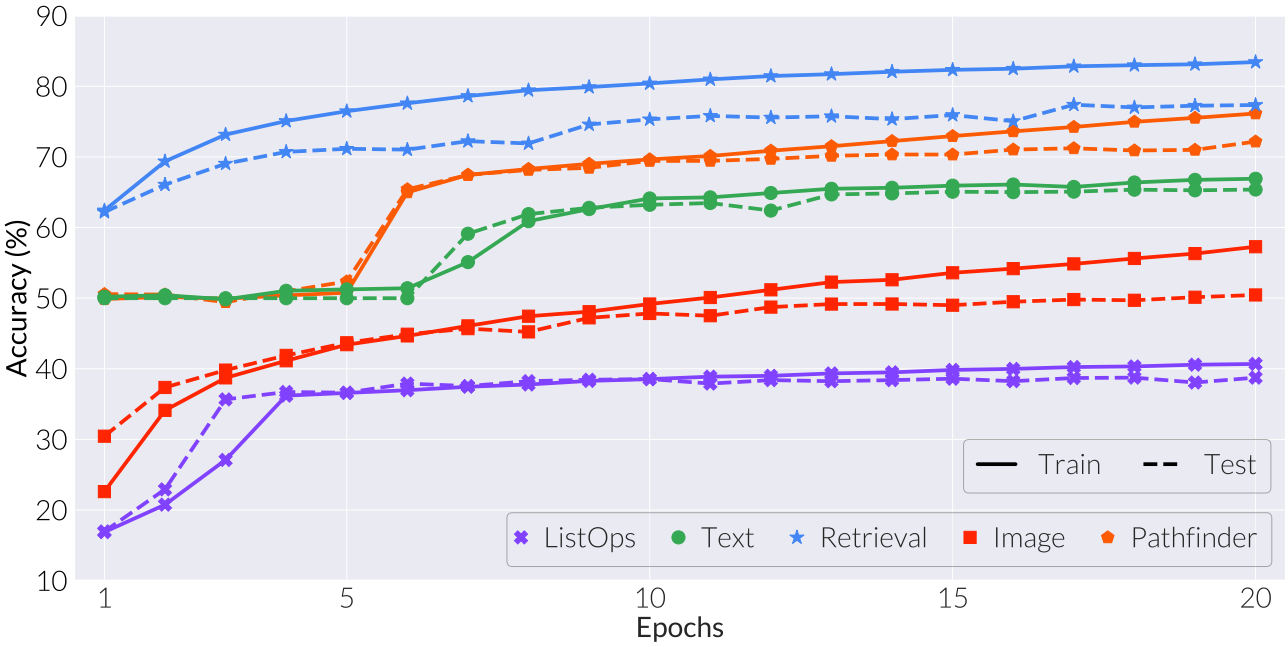

Figure 8: The learning curves of multi-layered Hrrformer in the LRA tasks. The training performance is solid lines and the test is dashed. Where prior works required 200 epochs of training, we can see that **20** epochs are sufficient for our Hrrformer. In most of the tasks, the 10-epoch performance of our Hrrformer is still highly competitive.

C. Weight Visualization

The weight vector $\mathbf{w}$ is visualized for LRA image classification task. In this task, grayscale images of the CIFAR-10 dataset of dimension 32×32 are reshaped into a sequence of length 1024. Therefore, the weight vector has the shape of $\mathbb{R}^{1024\times 1}$. This vector is reshaped back to 32×32 for visualization which shows where in the image the weight vector of each head

Table 6: Inference timing comparison between single Hrrformer block and single Transformer block for different batch sizes (2-32). The experiment is performed on the LRA text classification task.

Batch Size	Hrrformer		Transformer	
	Time (s)	Memory (MB)	Time (s)	Memory (MB)
2	152.99	663.88	975.98	1584.53
3	127.34	936.51	815.30	4809.95
4	118.39	938.61	813.72	4809.95
5	117.15	1475.48	812.09	9104.92
6	115.37	1481.77	810.57	9107.01
7	115.44	1483.87	810.14	9109.11
8	113.01	1488.06	810.59	9109.11
9	114.81	2563.90	809.61	17701.14
10	113.34	2563.90	809.87	17701.14
11	113.83	2570.19	808.71	17705.34
12	113.11	2572.29	808.52	17705.34
13	114.65	2576.48	808.35	17707.43
14	114.64	2578.58	808.66	17709.53
15	114.42	2582.77	808.12	17711.63
16	113.81	2589.07	808.80	17711.63
17	86.80	2593.26	807.34	30976.11
18	85.95	4742.84	806.94	30976.11
19	85.56	4749.13	806.91	30978.21
20	85.11	4749.13	808.78	30980.31
21	84.78	4755.42	806.70	30980.31
22	83.95	4757.52	806.70	30982.41
23	83.23	4763.81	806.50	30986.60
24	81.84	4765.91	807.04	30988.70
25	83.06	4768.01	809.12	30988.70
26	83.01	4772.20	806.10	30990.79
27	82.87	4776.39	806.89	30992.89
28	82.70	4780.59	806.70	30994.99
29	82.60	4784.78	807.45	30994.99
30	82.30	4788.98	806.71	30999.18
31	82.44	4791.07	807.51	30999.18
32	80.83	4797.37	807.13	31001.28

Table 7: Inference time comparison with different self-attention models. The experiment is performed on the LRA text classification task with 6 layers of the encoder. In the chart, * indicates single layer.

Model	Time (s) ↓	Speed (examples per second) ↑	Memory (MB) ↓
Local Attention	1910.33	13.09	9369.16
Synthesizer	1848.77	13.52	8983.28
Sinkhorn Transformer	1848.76	13.52	8983.28
Transformer	813.67	30.72	4805.75
Sparse Transformer	361.69	69.12	5229.38
Longformer	337.81	74.01	2815.56
Performer	170.75	146.41	728.89
Linear Transformer	163.15	153.23	913.44
BigBird	92.89	269.14	645.01
Linformer	88.96	281.03	645.01
Hrrformer	33.38	748.95	**527.56**
Hrrformer*	**31.82**	**785.67**	**527.56**

puts its attention. Figure 9 demonstrates the attention map of the 4 heads in each of the 3 layers of Hrrformer for all the CIFAR-10 classes.

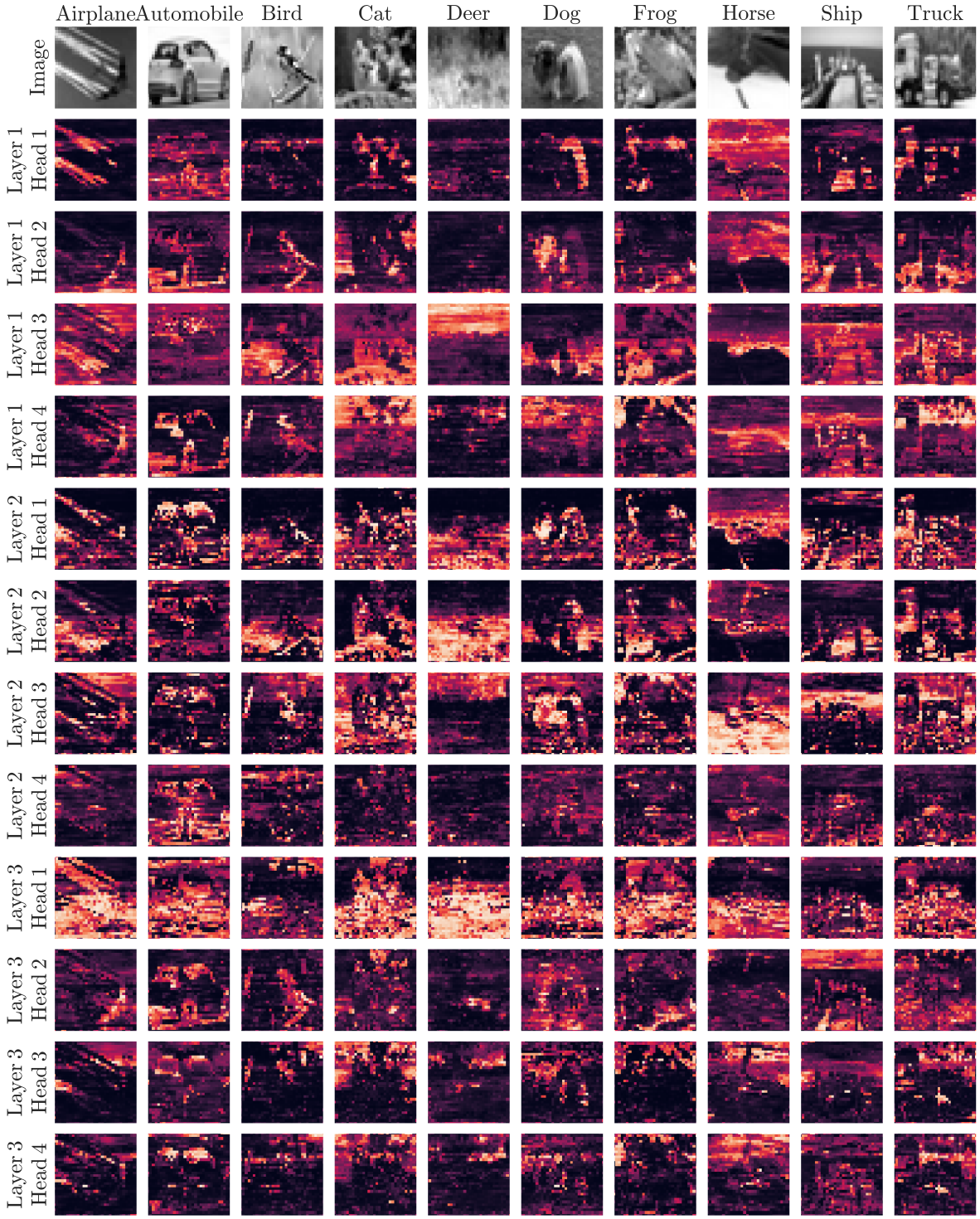

Figure 9: Visualization of weight $\mathbf{w} \in \mathbb{R}^{1024\times 1}$ vector of multi-layer Hrrformer, reshaped to 32×32, the shape of the original image of the CIFAR-10 dataset used in the LRA image classification task.

For the standard Transformer, the responses are a matrix of cross-correlations rather than a single vector. This makes the response more difficult to interpret. To visualize in the same manner we average the response of correlations with respect to a single item t to get the same 1024 shape, and visualize the results in Figure 10. As can be seen, the identification of structure is not as obviously.

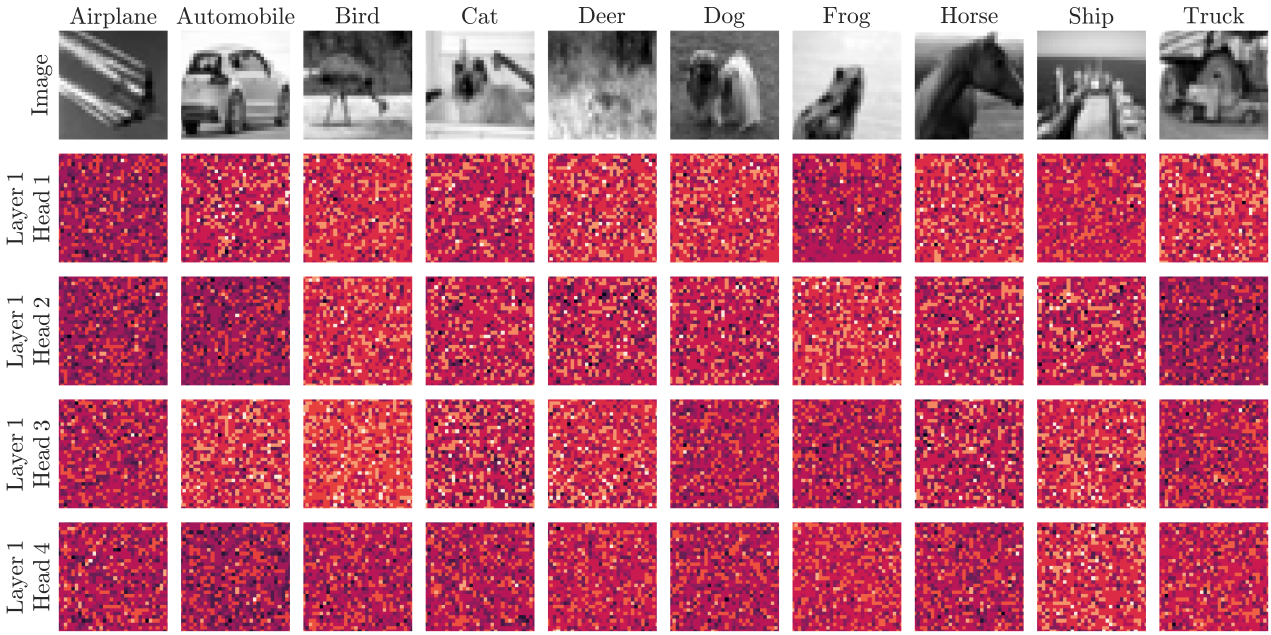

Figure 10: Visualization of transformer's weight matrix of shape 1024×1024. The average attention across the key dimension is taken to reduce its dimension to 1024×1 and reshaped to 32×32, the shape of the original image of the CIFAR-10 dataset used in the LRA Image classification task.

D. How Softmax "Denoises" Dot Product

To understand how we can use the softmax operation as a kind of denoising step, consider the H dimensional vectors $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{z}$. If each element of all these vectors is sampled from $\mathcal{N}(0, 1/H)$, then we would expect that $(\boldsymbol{a}\otimes\boldsymbol{b}+\boldsymbol{c}\otimes\boldsymbol{d})^\top \boldsymbol{a}^\dagger \approx 1$. Similarly, the value $\boldsymbol{z}$ is not present, so we expect that $(\boldsymbol{a} \otimes \boldsymbol{b} + \boldsymbol{c} \otimes \boldsymbol{d})^\top \boldsymbol{z}^\dagger \approx 0$. Now let us consider our use case, where the I.I.D. property is not true, and the query that is a noisy version of a present item. For simplicity of notation, we will use the explicit case of $H = 2$ dimensions. We can query for $\boldsymbol{a} + \boldsymbol{z}$ get:

$$\frac{(a_0 + z_0)\,(a_0 b_0 + a_1 b_1 + c_0 d_0 + c_1 d_1) - (a_1 + z_1)\,(a_0 b_1 + a_1 b_0 + c_0 d_1 + c_1 d_0)}{(a_0 - a_1 + z_0 - z_1)\,(a_0 + a_1 + z_0 + z_1)}$$

Similarly if we query with $\boldsymbol{c} + \boldsymbol{z}$ we instead get:

$$\frac{(c_0 + z_0)\,(a_0 b_0 + a_1 b_1 + c_0 d_0 + c_1 d_1) - (c_1 + z_1)\,(a_0 b_1 + a_1 b_0 + c_0 d_1 + c_1 d_0)}{(c_0 - c_1 + z_0 - z_1)\,(c_0 + c_1 + z_0 + z_1)}$$

Notice that in both cases we have shared terms that are multiplied and added together. Under the sufficient conditions of I.I.D Gaussian, the linearity of expectation results in these terms canceling out into a single random variable with a zero mean.

However, these also have the artifact in our application that for a non-present query, the response magnitude will have a similar value due to the repeated shared terms.

We can simplify our understanding of this by imagining that there is an additional noise constant ϵ that we must add to each noise term. Then when we apply the softmax operation, we obtain the benefit that the softmax function is invariant to constant shifts in the input, i.e., $\forall \epsilon \in \mathbb{R}, \mathrm{softmax}(\boldsymbol{x} + \epsilon) = \mathrm{softmax}(\boldsymbol{x})$. Thus, we get the practical effect of softmax removing noise that we incur for not using I.I.D. Gaussian as the elements of our vectors.