`MonarchAttention`: Zero-Shot Conversion to Fast, Hardware-Aware Structured Attention

Can Yaras
University of Michigan
`cjyaras@umich.edu`

Alec S. Xu
University of Michigan
`alecx@umich.edu`

Pierre Abillama
University of Michigan
`pabillam@umich.edu`

Changwoo Lee
University of Michigan
`cwoolee@umich.edu`

Laura Balzano
University of Michigan
`girasole@umich.edu`

Abstract

Transformers have achieved state-of-the-art performance across various tasks, but suffer from a notable quadratic complexity in sequence length due to the attention mechanism. In this work, we propose `MonarchAttention` – a novel approach to sub-quadratic attention approximation via Monarch matrices, an expressive class of structured matrices. Based on the variational form of softmax, we describe an efficient optimization-based algorithm to compute an approximate projection of softmax attention onto the class of Monarch matrices with $\Theta(N\sqrt{N}d)$ computational complexity and $\Theta(Nd)$ memory/IO complexity. Unlike previous approaches, `MonarchAttention` is both (1) transferable, yielding minimal performance loss with no additional training, even when replacing every attention layer of the Transformer, and (2) hardware-efficient, utilizing the highest-throughput tensor core units on modern GPUs. With optimized kernels, `MonarchAttention` achieves substantial speed-ups in wall-time over `FlashAttention-2`: $1.4\times$ for shorter sequences $(N = 256)$, $4.5\times$ for medium-length sequences $(N = 4K)$, and $8.2\times$ for longer sequences $(N = 16K)$. We demonstrate the quality of `MonarchAttention` on diverse tasks and architectures in vision and language problems, showing that it flexibly and accurately approximates softmax attention in a variety of contexts. Our code is available at `https://github.com/cjyaras/monarch-attention`.

1 Introduction

Over the past decade, Transformers (Vaswani et al., 2017) have become the dominant architecture for generating and processing various data modalities, such as text (Brown et al., 2020), images (Dosovitskiy et al., 2021), and speech (Radford et al., 2023). Central to the Transformer's success is *attention*, the mechanism through which complex interactions within sequential data are captured through weighted combinations of embeddings at every position in the sequence. Famously, the attention mechanism has a quadratic-time complexity $\Theta(N^2d)$ in the length of the sequence N, where d is the head dimension, which is a key bottleneck for both training and inference, particularly in long sequence problems. To address this, numerous works have proposed sub-quadratic substitutes for attention. Yet, such approaches are either (1) not transferable, requiring training from scratch or fine-tuning of existing models, or (2) do not yield speed-ups in practice (except on extremely long sequences) due to a gap between theoretical complexity and practical considerations for modern GPUs, especially compared to highly optimized implementations (Dao et al., 2022b).

In this work, we propose `MonarchAttention`: a novel sub-quadratic attention substitute

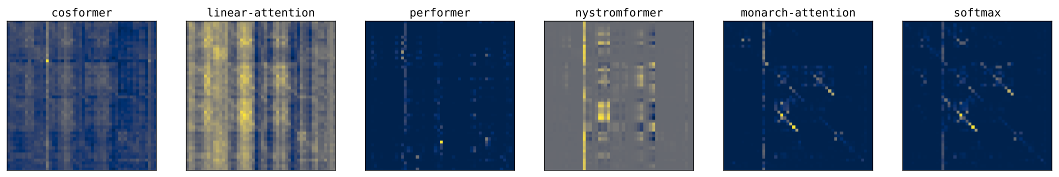


Figure 1: **Approximation of softmax attention via `MonarchAttention`.** By directly optimizing the softmax variational objective constrained to Monarch matrices, `MonarchAttention` yields accurate zero-shot approximation to softmax attention compared to other hardware-friendly, efficient attention baselines. Attention maps extracted from RoBERTa on the SQuAD dataset in Section 4.

based on approximating the attention matrix via *Monarch* matrices (Dao et al., 2022a), a class of expressive structured matrices. At first glance, this is computationally infeasible – for sequence length N, computing an exact projection onto the set of Monarch matrices has a super-quadratic $O(N^2\sqrt{N})$-time complexity, not to mention that we need to form the entire $N \times N$ attention matrix. Instead, we reframe the computation of the attention matrix as an optimization problem in terms of the variational form of softmax, and exploit low-dimensional structure in the variational objective when constrained to the set of Monarch matrices – this yields a sub-quadratic $\Theta(N\sqrt{N}d)$-time approximation, where d is the head dimension. This approach combines two optimization-based improvements for its success: the variational form of the softmax nonlinearity with a structured factorization approach analogous to those for low-rank approximation of a matrix (Chi et al., 2019), where rather than computing a full SVD and truncating to the desired rank, one can more efficiently minimize a Frobenius norm objective constrained to the set of low-rank matrices. We briefly review prior work on structured matrices, including Monarch matrices, as well as existing approaches to efficient attention.

Structured & Monarch Matrices. We use the phrase "structured matrices" to mean those that admit sub-quadratic storage and matrix-vector multiplication, such as low-rank or sparse matrices. There are many useful classes of structured matrices, such as those with *low displacement rank* (Kailath et al., 1979), which includes Toeplitz, Hankel, Vandermonde, Cauchy matrices (Pan, 2001); *orthogonal polynomial transforms* (Chihara, 2014), which includes discrete Fourier/cosine and Hadamard transforms; *butterfly factorizations* (Dao et al., 2019), which implement fast matrix-vector multiplication via a recursive divide-and-conquer algorithm similar to that of fast Fourier transforms (FFTs); and *Monarch matrices*, an expressive family of structured matrices (generalizing butterfly matrices and thereby many fast transforms) that overcome unfavorable memory access patterns typical to FFT-like algorithms by implementing matrix products via batched dense matrix multiplications (also called matmuls) on fast tensor cores found in modern GPUs.

Sub-Quadratic Attention. Nearly all approaches to sub-quadratic attention approximate the attention matrix by a structured matrix, specifically low-rank and/or sparse.

- **Low-Rank.** Motivated by Johnson-Lindenstrauss embeddings, Wang et al. (2020) propose sketching the key and value matrices along the sequence dimension via learnable projections. Katharopoulos et al. (2020) introduce *linear attention*, where the exponential kernel is approximated via inner products of queries and keys lifted via some feature map. Several follow-up works proposed various feature maps, such as the exponential linear unit (ELU) (Katharopoulos et al., 2020), random positive features (Choromanski et al., 2021), rectified linear unit (ReLU) with cosine reweighting (Qin et al., 2022), and learnable single-layer multi-layer perceptrons (MLPs) (Zhang et al., 2024). Xiong et al. (2021) use the Nyström method for computing low-rank approximations by sampling rows and columns.
- **Sparse.** Child et al. (2019) introduce sparsity by applying fixed, structured sparse masks on the attention matrix. In particular, Chen et al. (2022) propose a particular *block* butterfly matrix for the sparse mask, which is more hardware-friendly at the cost of reduced expressiveness. Those that do not enforce a structure on the sparsity pattern include Kitaev et al. (2020); Daras et al. (2020) where they utilize locality-sensitive hashing (LSH) on shared query/key vectors to only compute attention within clusters of similar tokens.

- **Low-Rank + Sparse.** Inspired by robust PCA, Chen et al. (2021) decompose the attention matrix into a sum of two matrices: an unstructured sparse component using LSH and a low-rank component that is constructed via linear attention. Han et al. (2024) propose to subsample columns of the non-normalized attention matrix based on row norms of the value matrix, while estimating the softmax normalization factors from a few large elements via LSH.

We note that there are significant drawbacks to the approaches described above. Pure low-rank methods are often fast and hardware-friendly, but are typically not suitable as drop-in replacements for attention in pre-trained Transformers due to the prevalence of "strongly diagonal", high-rank attention matrices where attention weights are concentrated locally in a sequence. Making up for this with a fixed sparsity pattern does not allow for data-dependent support of the attention matrix, necessary for zero-shot conversion. Finally, sparsity/LSH-based approaches that do not have a fixed sparsity pattern improve on accuracy over low-rank approximations but suffer from significant overhead due to GPU incompatibility. `MonarchAttention`, on the other hand, achieves the best of both worlds: it is fast and hardware-friendly due to utilization of tensor cores for batched matmuls, while computing highly accurate approximations to the extent that it can directly replace softmax attention with no additional training.

We conclude this section by discussing closely related works. Dao et al. (2022b) propose `FlashAttention`, an IO-aware streaming algorithm for computing exact softmax attention. We show in Section 3 that each step of `MonarchAttention` can be written as a `FlashAttention`-like computation, allowing for similar IO savings to `FlashAttention` – in fact, we demonstrate that `MonarchAttention` achieves a strictly better worst-case IO complexity compared to `FlashAttention`. We also note that Dao et al. (2022b) propose to further accelerate `FlashAttention` using block butterfly attention masks, so `MonarchAttention` can be viewed as a generalization of block-sparse `FlashAttention` to more general Monarch matrices. Finally, `MonarchAttention` is closely related to Monarch Mixer (Fu et al., 2023), a mixer-type architecture (Tolstikhin et al., 2021) that utilizes Monarch instead of dense matrices for token and channel mixing. `MonarchAttention` also uses Monarch matrices for mixing tokens – however, it is based on the attention operation which is *data-dependent*, unlike Monarch Mixer.

Organization. In Section 2, we discuss preliminaries on (softmax) attention, and Monarch matrices. In Section 3, we describe the `MonarchAttention` algorithm and implementation. In Section 4, we evaluate `MonarchAttention` in a variety of settings for zero-shot conversion to sub-quadratic attention and benchmark its implementation. In Section 5, we discuss limitations and future directions.

2 Preliminaries

Notation. We use $[N]$ to denote the index set $\{1, 2, \ldots, N\}$. We use Δ^N to denote the $(N-1)$ dimensional unit simplex, given by $\Delta^N = \{\boldsymbol{a} \in \mathbb{R}^N : \boldsymbol{a} \succeq 0, \langle \mathbf{1}_N, \boldsymbol{a} \rangle = 1\}$. We denote the $m \times m$ identity matrix by $\boldsymbol{I}_m$. We use the notation $\boldsymbol{A}_{ijk}$ to denote an element of a 3-way tensor, and $\boldsymbol{A}_{i,:,k}$ to denote a slice. We use δ_{kl} to denote the Kronecker delta that is 1 if $k = l$ and otherwise 0.

Softmax. The softmax function $\mathbb{R}^N \to \Delta^N$ maps N real numbers to the $(N-1)$-dimensional unit simplex, and is defined as

$$[\text{softmax}(\boldsymbol{z})]_i := \frac{\exp(\boldsymbol{z}_i)}{\sum_j \exp(\boldsymbol{z}_j)}, \ \forall i \in [N]. \tag{1}$$

An alternative definition (Blondel et al., 2019) is given by the following variational form:

$$\text{softmax}(\boldsymbol{z}) := \underset{\boldsymbol{a} \in \Delta^N}{\arg\max}\ \langle \boldsymbol{a}, \boldsymbol{z} \rangle + H(\boldsymbol{a}), \tag{2}$$

where $H(\boldsymbol{a}) = -\sum_i \boldsymbol{a}_i \log \boldsymbol{a}_i$ is Shannon entropy. See Appendix A for equivalence of (1) and (2).

Attention. Given query, key, value matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N \times d}$, where N is the sequence length and d is the head dimension, a single head of standard softmax attention[1] computes

$$\boldsymbol{O} = \text{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top\right) \boldsymbol{V}, \tag{3}$$

[1]Typically, the $\boldsymbol{Q}\boldsymbol{K}^\top$ matrix is scaled by a factor of $d^{-1/2}$, but this can be absorbed into $\boldsymbol{Q}$.

where the softmax function is applied across rows. The computational complexity of attention is $\Theta(N^2 d)$ for each forward pass, because the matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are data-dependent.

Monarch Matrices. Given $N = m \times b$ for integers m, b, we define a block rank-one matrix $\boldsymbol{B} \in \mathbb{R}^{N \times N}$ as

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_{11} & \dots & \boldsymbol{B}_{1m} \\ \vdots & \ddots & \vdots \\ \boldsymbol{B}_{b1} & \dots & \boldsymbol{B}_{bm} \end{bmatrix}, \quad \text{where} \quad \boldsymbol{B}_{jk} = \boldsymbol{L}_{jk} \boldsymbol{R}_{kj}^\top \in \mathbb{R}^{m \times b}$$

for some $\boldsymbol{L}_{jk} \in \mathbb{R}^m, \boldsymbol{R}_{kj} \in \mathbb{R}^b$ for $j \in [b]$ and $k \in [m]$. It follows that

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{L}_1^\top & & & \\ & \boldsymbol{L}_2^\top & & \\ & & \ddots & \\ & & & \boldsymbol{L}_b^\top \end{bmatrix} \boldsymbol{P} \begin{bmatrix} \boldsymbol{R}_1 & & & \\ & \boldsymbol{R}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{R}_m \end{bmatrix},$$

where $\boldsymbol{L}_j \in \mathbb{R}^{m \times m}$ for $j \in [b]$ and $\boldsymbol{R}_k \in \mathbb{R}^{b \times b}$ for $k \in [m]$, and $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ is a "transpose"[2] permutation matrix whose $(i+1)$-th row is given by $\boldsymbol{e}_{\sigma(i)+1}$ where

$$\sigma(i) = b \cdot (i \bmod m) + \left\lfloor \frac{i}{m} \right\rfloor, \quad i \in \{0, \dots, N-1\}.$$

Given the above, a Monarch matrix $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ is given by $\boldsymbol{M} = \boldsymbol{P}^\top \boldsymbol{B}$ – in other words, it is a row-permuted block rank-one matrix. When $m = b = \sqrt{N}$, storing such a matrix requires only $\Theta(N\sqrt{N})$ space, while matrix multiplication (matmul) with a matrix $\boldsymbol{V} \in \mathbb{R}^{N \times d}$ can be computed efficiently in $\Theta(N\sqrt{N}d)$ operations (as opposed to $\Theta(N^2 d)$ for dense matrices) with batched matmuls and transposes:

$$\boldsymbol{MV} = \boldsymbol{O}, \quad \text{where} \quad \boldsymbol{O}_{b\cdot(l-1)+j,v} = \sum_k \boldsymbol{L}_{jkl} \boldsymbol{Y}_{jkv}, \quad \boldsymbol{Y}_{jkv} = \sum_i \boldsymbol{R}_{kji} \boldsymbol{V}_{b\cdot(k-1)+i,v} \tag{4}$$

for $i, j \in [b], k, l \in [m]$. A useful characterization of $\boldsymbol{M}$ is in block form:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_{11} & \dots & \boldsymbol{M}_{1m} \\ \vdots & \ddots & \vdots \\ \boldsymbol{M}_{m1} & \dots & \boldsymbol{M}_{mm} \end{bmatrix}, \tag{5}$$

where $\boldsymbol{M}_{lk} \in \mathbb{R}^{b \times b},\ [\boldsymbol{M}_{lk}]_{ji} = \boldsymbol{L}_{jkl} \boldsymbol{R}_{kji},\ \forall i, j \in [b], k, l \in [m]$.

3 MonarchAttention

The main goal of `MonarchAttention` is to find a Monarch matrix $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ in $o(N^2 d)$ time such that $\boldsymbol{M} \approx \mathrm{softmax}(\boldsymbol{QK}^\top)$. Then, we can approximately compute the output $\boldsymbol{O} = \boldsymbol{MV}$ using efficient matmul. We can do this by viewing the softmax operation as an optimization problem via its variational form (2), whose objective can be efficiently maximized with exact alternating steps when constrained to Monarch matrices. As shown in Figure 1, this yields highly accurate approximations to the softmax attention matrix.

Softmax Objective. First, from (2) we can write

$$\sigma(\boldsymbol{QK}^\top) = \operatorname*{arg\,max}_{\boldsymbol{A} \in \Delta^{N \times N}} f(\boldsymbol{A}; \boldsymbol{Q}, \boldsymbol{K}) := \langle \boldsymbol{A}, \boldsymbol{QK}^\top \rangle + H(\boldsymbol{A}), \tag{6}$$

where $\Delta^{N \times N}$ denotes a matrix whose rows lie on Δ^N, and $H(\boldsymbol{A}) = -\sum_{i,j} \boldsymbol{A}_{ij} \log \boldsymbol{A}_{ij}$. Here we demonstrate how to handle this objective function when $\boldsymbol{A}$ has Monarch structure, and defer discussion of the simplex constraint to the next paragraph. For a dense matrix $\boldsymbol{A}$, computing $f(\boldsymbol{A}; \boldsymbol{Q}, \boldsymbol{K})$

[2] $\boldsymbol{Px}$ corresponds to row-major reshaping $\boldsymbol{x} \in \mathbb{R}^N$ to $\mathbb{R}^{m \times b}$, transposing to $\mathbb{R}^{b \times m}$, then row-major flattening back to $\mathbb{R}^N$. See Appendix B for an example.

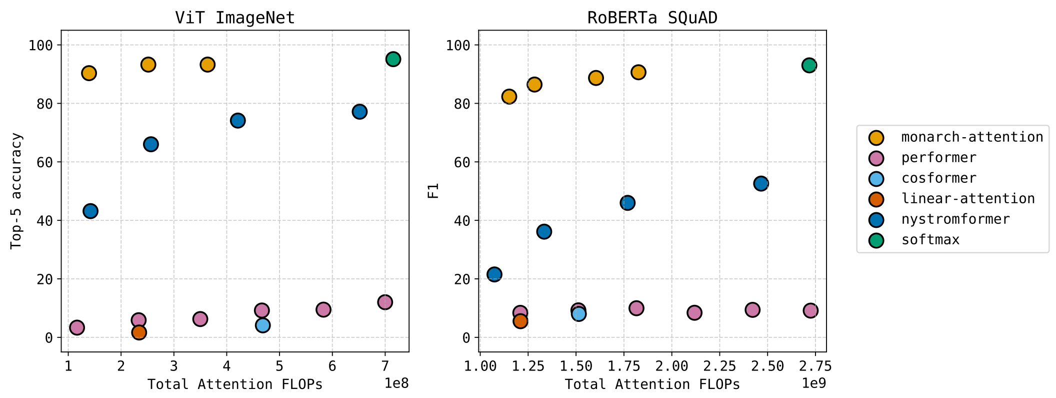


Figure 2: **Zero-shot conversion of attention layers for image classification and question answering.** We vary hyperparameters for various baselines to evaluate model quality vs compute tradeoff. *Left.* Top-5 accuracy vs. total attention FLOPs across all layers for ViT on ImageNet. *Right.* F1 score vs total attention FLOPs across all layers for RoBERTa on SQuAD.

requires $\Theta(N^2 d)$ operations, which is the same as computing $\sigma(\boldsymbol{Q}\boldsymbol{K}^\top)$ directly. However, when $\boldsymbol{A}$ is a Monarch matrix $\boldsymbol{M} = \boldsymbol{P}^\top \boldsymbol{B}$, we have

$$\begin{aligned} f(\boldsymbol{P}^\top \boldsymbol{B}; \boldsymbol{Q}, \boldsymbol{K}) &= \langle \boldsymbol{P}^\top \boldsymbol{B}, \boldsymbol{Q}\boldsymbol{K}^\top \rangle + H(\boldsymbol{P}^\top \boldsymbol{B}) \\ &= \langle \boldsymbol{B}, \widetilde{\boldsymbol{Q}}\boldsymbol{K}^\top \rangle + H(\boldsymbol{B}) = \sum_{j,k} f(\boldsymbol{B}_{jk}; \widetilde{\boldsymbol{Q}}_j, \boldsymbol{K}_k), \end{aligned}$$

where $\widetilde{\boldsymbol{Q}} = \boldsymbol{P}\boldsymbol{Q}$, and $\widetilde{\boldsymbol{Q}}_j \in \mathbb{R}^{m \times d}, \boldsymbol{K}_k \in \mathbb{R}^{b \times d}$ are the j-th and k-th block of rows of $\widetilde{\boldsymbol{Q}}, \boldsymbol{K}$ respectively. Then, for each $j \in [b], k \in [m]$ we evaluate f on the rank-one matrix $\boldsymbol{B}_{jk} = \boldsymbol{L}_{jk}\boldsymbol{R}_{kj}^\top$:

$$\begin{aligned} f(\boldsymbol{B}_{jk}; \widetilde{\boldsymbol{Q}}_j, \boldsymbol{K}_k) &= \langle \boldsymbol{L}_{jk}\boldsymbol{R}_{kj}^\top, \widetilde{\boldsymbol{Q}}_j \boldsymbol{K}_k^\top \rangle - \sum_{l,i} \boldsymbol{L}_{jkl}\boldsymbol{R}_{kji} \log(\boldsymbol{L}_{jkl}\boldsymbol{R}_{kji}) \\ &= \langle \widetilde{\boldsymbol{Q}}_j^\top \boldsymbol{L}_{jk}, \boldsymbol{K}_k^\top \boldsymbol{R}_{kj} \rangle - \sum_{l,i} \boldsymbol{L}_{jkl}\boldsymbol{R}_{kji} \log \boldsymbol{L}_{jkl} - \sum_{l,i} \boldsymbol{L}_{jkl}\boldsymbol{R}_{kji} \log \boldsymbol{R}_{kji} \\ &= \langle \widetilde{\boldsymbol{Q}}_j^\top \boldsymbol{L}_{jk}, \boldsymbol{K}_k^\top \boldsymbol{R}_{kj} \rangle + \left(\boldsymbol{1}^\top \boldsymbol{R}_{kj}\right) \cdot H(\boldsymbol{L}_{jk}) + \left(\boldsymbol{1}^\top \boldsymbol{L}_{jk}\right) \cdot H(\boldsymbol{R}_{kj}). \end{aligned}$$

Thus, for each $j \in [b], k \in [m]$ we only need $\Theta((m+b)d)$ operations to compute $f(\boldsymbol{B}_{jk}; \widetilde{\boldsymbol{Q}}_j, \boldsymbol{K}_k)$ due to $\widetilde{\boldsymbol{Q}}_j^\top \boldsymbol{L}_{jk}$ and $\boldsymbol{K}_k^\top \boldsymbol{R}_{kj}$. We emphasize that the rank-one structure implies separability of the entropy term, meaning we can compute the entropy on $\boldsymbol{L}_{jk}$ and $\boldsymbol{R}_{kj}$ individually and avoid the need to materialize $\boldsymbol{B}_{jk}$, which would incur $\Theta(mb)$ cost as opposed to $\Theta(m+b)$. Since there are $m \cdot b$ many $\boldsymbol{B}_{jk}$ matrices, we have in total $\Theta((m^2 b + b^2 m)d)$ operations to compute $f(\boldsymbol{M}; \boldsymbol{Q}, \boldsymbol{K})$, which for $m = b = \sqrt{N}$ is $\Theta(N\sqrt{N}d)$, improving on the dense computation by a factor of $\sqrt{N}$.

Alternating Maximization with Constraints. We will now explain the alternating maximization approach for optimizing f. When $\boldsymbol{L}$ is fixed, the objective is concave in $\boldsymbol{R}$, and vice-versa – therefore, we can derive closed form expressions via KKT conditions for $\boldsymbol{L}$ and $\boldsymbol{R}$ that maximize f with one of $\boldsymbol{L}$ or $\boldsymbol{R}$ fixed, which will constitute a single update step. Evaluating (and therefore differentiating) f w.r.t. $\boldsymbol{L}$ and $\boldsymbol{R}$ can be done in $\Theta(N\sqrt{N}d)$ time, which will be the same complexity as one of these steps. For T steps, this will require $\Theta(TN\sqrt{N}d)$ computation; provided that $T = o(\sqrt{N})$, this will still be sub-quadratic. However, the constraint $\boldsymbol{M} \in \Delta^{N \times N}$ presents a challenge in its current form, since this requires materializing $\boldsymbol{M}$ to check that each entry is non-negative. Instead, we use the fact that

$$\boldsymbol{L}_{j,:,l} \in \Delta^m,\ \boldsymbol{R}_{kj} \in \Delta^b,\ \forall j \in [b], \forall k, l \in [m] \implies \boldsymbol{M} \in \Delta^{N \times N},$$

i.e., slices of $\boldsymbol{L}, \boldsymbol{R}$ individually lying on the unit simplex is sufficient to enforce the constraint on $\boldsymbol{M}$. This is easily seen from (5) – obviously if $\boldsymbol{L}_{jkl}, \boldsymbol{R}_{kji} \geq 0$, then $[\boldsymbol{M}_{lk}]_{ji} \geq 0$. Moreover, this also

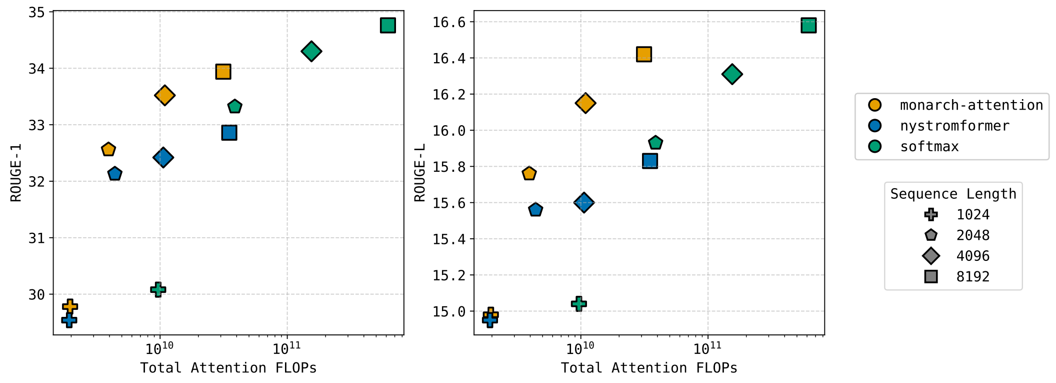


Figure 3: **Zero-shot conversion of attention layers for long sequence summarization.** We vary the sequence length of the text to be summarized to evaluate model quality vs compute tradeoff. We report recall-based ROUGE-1 and ROUGE-L scores vs. total attention FLOPs across all layers for BART on BookSum-chapters.

enforces the sum-to-one constraint, as rows of $\boldsymbol{M}$ sum as

$$\sum_{k,i}[\boldsymbol{M}_{lk}]_{ji} = \left(\sum_k \boldsymbol{L}_{jkl}\right)\left(\sum_i \boldsymbol{R}_{kji}\right) = 1.$$

We now present the updates for $\boldsymbol{L}, \boldsymbol{R}$. Initializing $\boldsymbol{L}^{(0)}_{jkl} = \delta_{kl}$ as identity, we have

$$\boldsymbol{R}^{(t)} = \mathrm{softmax}_i(\boldsymbol{Z}^{(t)}_R), \quad \boldsymbol{Z}^{(t)}_{R,kji} = \boldsymbol{\beta}^{(t)}_{R,kji}/\boldsymbol{c}^{(t)}_{R,kj}, \tag{7}$$

$$\boldsymbol{L}^{(t)} = \mathrm{softmax}_k(\boldsymbol{Z}^{(t)}_L), \quad \boldsymbol{Z}^{(t)}_{L,jkl} = \boldsymbol{\beta}^{(t)}_{L,jkl} - \boldsymbol{c}^{(t)}_{L,jk}, \tag{8}$$

for $t \in [T]$, where $\mathrm{softmax}_k, \mathrm{softmax}_i$ are applied along k and i index dimensions respectively, and

$$\boldsymbol{\beta}^{(t)}_{R,kji} = \sum_v \boldsymbol{\alpha}^{(t)}_{R,kjv}\overline{\boldsymbol{K}}_{kiv}, \quad \boldsymbol{\alpha}^{(t)}_{R,kjv} = \sum_l \boldsymbol{L}^{(t-1)}_{jkl}\overline{\boldsymbol{Q}}_{jlv}, \quad \boldsymbol{c}^{(t)}_{R,kj} = \sum_l \boldsymbol{L}^{(t-1)}_{jkl}, \tag{9}$$

$$\boldsymbol{\beta}^{(t)}_{L,jkl} = \sum_v \boldsymbol{\alpha}^{(t)}_{L,jkv}\overline{\boldsymbol{Q}}_{jlv}, \quad \boldsymbol{\alpha}^{(t)}_{L,jkv} = \sum_i \boldsymbol{R}^{(t)}_{kji}\overline{\boldsymbol{K}}_{kiv}, \quad \boldsymbol{c}^{(t)}_{L,jk} = \sum_i \boldsymbol{R}^{(t)}_{kji}\log \boldsymbol{R}^{(t)}_{kji}, \tag{10}$$

where $\overline{\boldsymbol{Q}}_{jl}, \overline{\boldsymbol{K}}_{ki} \in \mathbb{R}^d$ are the $(b\cdot(l-1)+j)$-th and $(b\cdot(k-1)+i)$-th rows of $\boldsymbol{Q}$ and $\boldsymbol{K}$ respectively. The full derivation is provided in Appendix C.1. After T steps, we obtain the final Monarch approximation $\boldsymbol{M}^{(T)} \approx \sigma(\boldsymbol{Q}\boldsymbol{K}^\top)$ with factors $\boldsymbol{L}^{(T)}$ and $\boldsymbol{R}^{(T)}$, from which we output $\boldsymbol{M}^{(T)}\boldsymbol{V}$ using (4). A naïve implementation of the full algorithm is provided in Appendix C.2. We discuss in Appendix C.3 how padding can be incorporated into `MonarchAttention` for when N is not divisible by b. We empirically demonstrate the fast convergence of `MonarchAttention` in Appendix E.1.

Implementation. To minimize data movement and memory usage on GPU, we do not materialize $\boldsymbol{L}$ or $\boldsymbol{R}$ in high-bandwidth memory (HBM). In addition to $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{O}$, we only need to maintain states[3] $\boldsymbol{\alpha}^{(t)}_R, \boldsymbol{\alpha}^{(t)}_L, \boldsymbol{c}^{(t)}_R, \boldsymbol{c}^{(t)}_L$ from (9) and (10), resulting in $\Theta(Nd)$ additional memory. All other intermediate values are only materialized in on-chip SRAM, fusing all operations between the $\boldsymbol{\alpha}$ (and similarly $\boldsymbol{c}$) variables. For instance, from the above update equations, the computation of $\boldsymbol{\alpha}^{(t)}_L$ from $\boldsymbol{\alpha}^{(t)}_R$ is given by

$$\boldsymbol{\alpha}^{(t)}_{L,jkv} = \mathrm{softmax}_i\left(\frac{\sum_v \boldsymbol{\alpha}^{(t)}_{R,kjv}\overline{\boldsymbol{K}}_{kiv}}{\boldsymbol{c}^{(t)}_{R,kj}}\right)\overline{\boldsymbol{K}}_{kiv},$$

[3]The $\boldsymbol{\alpha}$ and $\boldsymbol{c}$ variables can share the same memory location as those corresponding to $\boldsymbol{R}$ can be derived from $\boldsymbol{L}$ (and vice-versa).

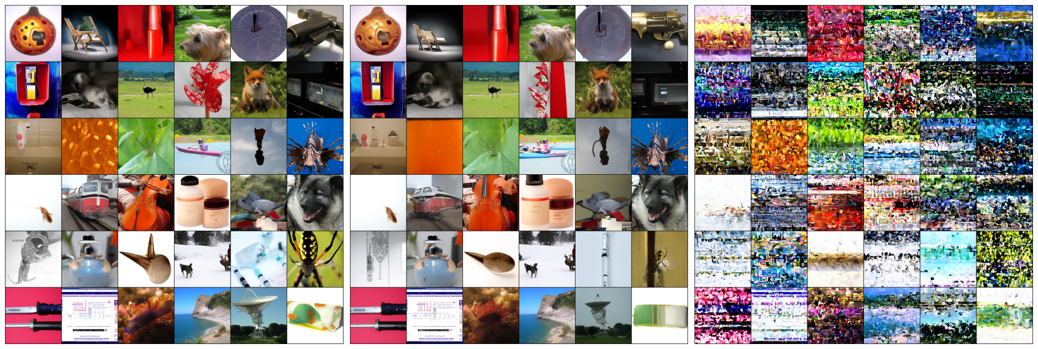
```python
def al_cl_kernel(aR, cR, Kb): # Computes aL, cL from aR, cR
    R = softmax(bmm(aR, Kb.transpose(1, 2)) / cR[:, :, None], dim=2)
    cL = sum(R * log(R), dim=2).transpose(0, 1)
    aL = bmm(R, Kb).transpose(0, 1)
    return aL, cL

def ar_cr_kernel(aL, cL, Qb): # Computes aR, cR from aL, cL
    L = softmax(bmm(aL, Qb.transpose(1, 2)) - cL[:, :, None], dim=1)
    cR = sum(L, dim=2).transpose(0, 1)
    aR = bmm(L, Qb).transpose(0, 1)
    return aR, cR

def al_y_cl_kernel(aR, cR, Kb, Vb): # Fuse al_cl_kernel + Monarch matmul 1st step
    R = softmax(bmm(aR, Kb.transpose(1, 2)) / cR[:, :, None], dim=2)
    cL = sum(R * log(R), dim=2).transpose(0, 1)
    aL = bmm(R, Kb).transpose(0, 1)
    y = bmm(R, Vb).transpose(0, 1)
    return aL, y, cL

def z_kernel(aL, y, cL, Qb): # Monarch matmul 2nd step
    L = softmax(bmm(Qb, aL.transpose(1, 2)) - cL[:, None, :], dim=2)
    z = bmm(L, y).transpose(0, 1)
    return z

def monarch_attention(Q, K, V, T): # Q, K, V: (N, d), T: number of steps
    Qb = Q.reshape(m, b, d).transpose(0, 1)
    Kb = K.reshape(m, b, d)
    Vb = V.reshape(m, b, d)
    aR = Q.reshape(m, b, d)
    cR = ones(m, b)

    for t in range(T-1):
        aL, cL = al_cl_kernel(aR, cR, Kb)
        aR, cR = ar_cr_kernel(aL, cL, Qb)

    aL, y, cL = al_y_cl_kernel(aR, cR, Kb, Vb)
    z = z_kernel(aL, y, cL, Qb)
    o = z.reshape(N, d)
    return o
```

Figure 4: **Python-like code for `MonarchAttention`.** Each kernel materializes all intermediate arrays in SRAM to reduce data movement.

which can be seen as a batched attention computation, meaning we can implement a `FlashAttention`-like kernel to reduce IO between HBM and SRAM. However, several aspects of this computation make it particularly IO-efficient. Besides the fact that $\overline{\boldsymbol{K}}$ acts as both the $\boldsymbol{K}$ and $\boldsymbol{V}$ matrices in (3), the effective sequence length is $\sqrt{N}$. This eliminates the need for tiling along the sequence length, except for very long sequences having $\Theta(\sqrt{N}d) > S$, where S is the size of on-chip SRAM. This means that we have an optimal IO complexity of $\Theta(Nd)$ for a single call, as opposed to the worst-case $O(N^2d^2/S)$ complexity of `FlashAttention`. The computation of $\boldsymbol{\alpha}_R^{(t)}$ from $\boldsymbol{\alpha}_L^{(t)}$, as well as the Monarch matmul, can be written in a similar fashion. Based on this, `MonarchAttention` not only achieves significant speed-up over `FlashAttention` for longer sequences, but also for shorter ones. Python-like code for `MonarchAttention` is given in Figure 4.

4 Experiments

In this section, we evaluate the zero-shot performance (no additional training) of `MonarchAttention` for converting pre-trained/fine-tuned Transformer attention layers to sub-quadratic attention in four different model/task settings. We compare with previous low-rank attention methods (Katharopoulos et al., 2020; Choromanski et al., 2021; Xiong et al., 2021; Qin et al., 2022); see Appendix D.1 for



(a) Softmax (b) `MonarchAttention` (c) Nyströmformer

Figure 5: **Visual quality of generated images for zero-shot conversion of attention layers.** Example images generated by with softmax (left), `MonarchAttention` (middle), and Nyströmformer (right). Only the first half of the attention layers of DiT are replaced.

more details on the baselines. We specifically exclude low-rank methods with learnable components (Wang et al., 2020; Zhang et al., 2024), since we are focused on the zero-shot setting, as well as sparsity/LSH-based approaches (Kitaev et al., 2020; Daras et al., 2020; Chen et al., 2021; Han et al., 2024), since these do not admit efficient implementations on current GPUs. We also benchmark our fast implementation of `MonarchAttention`, comparing with `FlashAttention-2` (Dao, 2024).

Image Classification with Vision Transformer. We convert all 12 attention layers, each having 12 heads with sequence length $N = 197$ and head dimension $d = 64$, of the 87M parameter ViT-B (Dosovitskiy et al., 2021) that has been pre-trained on ImageNet-21K (Deng et al., 2009) and fine-tuned on ImageNet-1K (Russakovsky et al., 2015) for image classification. To evaluate the performance at different FLOP counts, we vary the number of steps $T \in \{1, 2, 3\}$ for `MonarchAttention`, and vary the rank for Performer and Nyströmformer; see Appendix D.2 for more details on the set-up. The results are shown in the left panel of Figure 2. `MonarchAttention` achieves significant improvement over other baselines – compared to the original softmax attention, `MonarchAttention` loses only 5% accuracy to reduce attention FLOPs by 80%, or matches the performance to reduce attention FLOPs by 50%.

Question Answering with Encoder-Only Transformer. We convert the initial 4 and final 4 layers of the 12 attention layers, each having 12 heads with sequence length $N = 384$ and head dimension $d = 64$, of the 125M parameter RoBERTa-B (Liu et al., 2019) that has been pre-trained on a large English corpus and fine-tuned on SQuAD1.1 (Rajpurkar et al., 2016) for question answering. To evaluate the performance at different FLOP counts, we fix $T = 1$ and vary the block size b for `MonarchAttention`, and vary the rank for Performer and Nyströmformer; see Appendix D.3 for more details on the set-up. The results are shown in the right panel of Figure 2. Once again, `MonarchAttention` achieves significant improvement over other baselines – compared to the original softmax attention, `MonarchAttention` loses only 10 points in F1 score to reduce attention FLOPs by 60%, or matches the performance to reduce attention FLOPs by 35%. We additionally provide a per-layer ablation study in Appendix E.2 to understand the impact of converting different layers in this setting.

Summarization with Encoder-Decoder Transformer. We convert all 6 attention layers, each having 12 heads with head dimension $d = 64$, in the encoder of the 139M parameter BART-B (Lewis et al., 2020) that has been pre-trained on a large English corpus and fine-tuned on BookSum-chapters (Kryściński et al., 2022) for summarization. We only convert the encoder model and leave the decoder intact. To evaluate the benefits of sub-quadratic attention for processing longer sequences, we truncate the text to be summarized to various sequence lengths N for each method, with $T \in \{2, 3\}$ for `MonarchAttention` depending on the sequence length; see Appendix D.4 for more details on the set-up. The results are shown in Figure 3. We see that `MonarchAttention` achieves a strictly better ROUGE score (Lin, 2004) vs. FLOPs tradeoff than even softmax attention, due to accurate and efficient processing of longer sequences. In particular, the $N = 8192$ `MonarchAttention` model

Table 1: **Quantitative results for zero-shot conversion of attention layers for image generation.** We report FID and sFID (using the original softmax attention model as reference) of DiT when replacing all or half of the attention layers.

Layers Replaced	**Method**	**Total Attention FLOPs** (10^9)	**FID** ($\downarrow$)	**sFID** ($\downarrow$)
–	Softmax	8.46	–	–
All	Nyströmformer	3.30	5.97	13.47
	`MonarchAttention`	3.44	**2.82**	**5.09**
First Half	Nyströmformer	5.88	8.17	19.01
	`MonarchAttention`	5.95	**0.39**	**0.66**
Second Half	Nyströmformer	5.88	6.76	13.58
	`MonarchAttention`	5.95	**1.98**	**3.36**

improves on the $N = 2048$ softmax attention model by 0.75 on ROUGE-1 and 0.5 on ROUGE-L with slightly fewer FLOPs, while the $N = 8192$ Nyströmformer model with similar FLOPs does strictly worse than softmax. Although we are mainly focused on the zero-shot setting, we also investigate the post-swap fine-tuning performance and stability of `MonarchAttention` on this task in Appendix E.3.

Image Generation with Diffusion Transformer. We convert a subset of the 28 attention layers, each having 16 heads with sequence length $N = 256$ and head dimension $d = 72$, of the 675M parameter DiT-XL (Peebles and Xie, 2023) that has been trained on ImageNet (Deng et al., 2009). We consider replacing either all layers, the first 14 layers, or the last 14 layers, and fix $T = 3$ for `MonarchAttention`; see Appendix D.5 for more details on the set-up. Examples of generated images with each method for replacing the first 14 layers are shown in Figure 5, where `MonarchAttention` produces clear images resembling those of softmax attention, while the Nyströmformer ones are extremely noisy. We also quantitatively evaluate the (s)FID scores (Heusel et al., 2017) of `MonarchAttention` compared with Nyströmformer, using images generated with the original softmax attention model as reference – the results are reported in Table 1. `MonarchAttention` noticeably outperforms Nyströmformer with similar FLOP count. In particular, using `MonarchAttention` in the first half of the DiT layers results in extremely small FID and sFID from the softmax attention model's images, while reducing FLOPs by nearly 30%. We provide a more fine-grained version of Table 1 for `MonarchAttention` in Appendix E.4 to investigate the impact of replacing different quarters of the DiT model.

Node Classification with Graph Transformer. Beyond image and language tasks, we also evaluate the performance of `MonarchAttention` for converting attention in Graph Transformers on a graph node classification task in Appendix E.5.

Benchmarking `MonarchAttention`. Finally, we validate that the computational/IO complexity reduction achieved by `MonarchAttention` translates into actual speed-ups on the NVIDIA A40, a modern GPU. We implement the pseudo-code described in Section 3 as four separate Triton kernels and compare it against the fastest available implementations of `FlashAttention-2` – either the Triton implementation or PyTorch's `scaled_dot_product_attention`, which calls the CUDA backend for `FlashAttention-2`. Using a fixed batch size E, number of heads H, and head dimension d, we sweep the input sequence length N, generate random $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ matrices, and compare the run-times of `FlashAttention-2` and `MonarchAttention` (with $b = \sqrt{N}$ and $T = 1$) in Figure 6 (left). As sequence length increases, `MonarchAttention` consistently outperforms `FlashAttention-2`, notably achieving up to $8.2\times$ speed-up with $N = 16384$. To highlight gains for shorter sequences, we implement `MonarchAttention` as a single fully-fused Triton kernel, with a single thread block computing a single head. For fixed sequence length $N = 256$, number of heads H, and head dimension d, we sweep the batch size E and compare the run-time of the fully-fused `MonarchAttention` kernel against `FlashAttention-2` in Figure 6 (right). With smaller batch sizes, we have low utilization of hardware, since we compute a single head with a single thread block. However, as we increase the batch size, `MonarchAttention` achieves up to $1.4\times$ speed-up over `FlashAttention-2`.

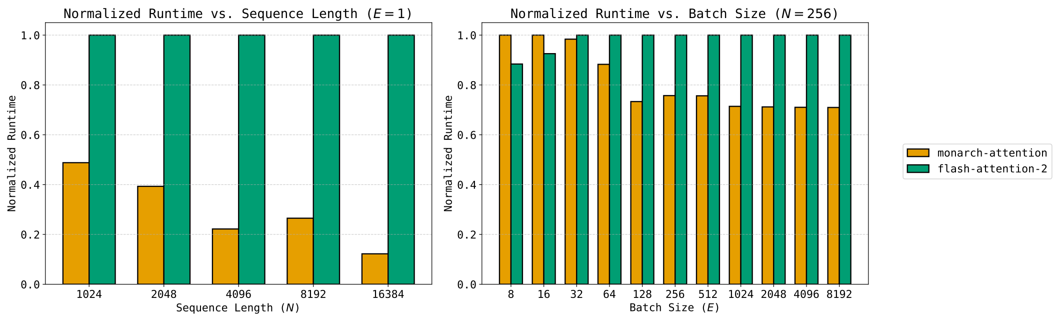


Figure 6: **Run-times of `MonarchAttention` and `FlashAttention-2` across various sequence lengths.** Normalized runtime ($1 = \text{slowest}, 0 = \text{fastest}$) of `MonarchAttention` and `FlashAttention-2` on NVIDIA A40 GPU. *Left*: sweep of sequence length N with $E = 1$, $H = 12$, and $d = 64$. *Right*: sweep of batch size E with $N = 256$, $H = 12$, and $d = 64$.

5 Conclusion

To conclude, we discuss several limitations and future directions for this work.

Autoregressive Transformers. In this paper, we have primarily focused on zero-shot approximation of attention in encoder-based Transformers, yet most generative language models of interest today are decoder-based autoregressive Transformers. One limitation of `MonarchAttention` is that it fundamentally cannot be applied to autoregressive generation, since there is no "attention matrix" to approximate – instead, single queries are passed through the attention mechanism in a streaming fashion, and discarded after each decode step. Instead, `MonarchAttention` could be used during prefill to efficiently compute keys/values for a provided prefix (e.g., long prompt or context). Moreover, `MonarchAttention` can be applied to emerging non-autoregressive paradigms such as diffusion language models (DLMs) to accelerate language generative models.

Accelerating Training. While we have presented `MonarchAttention` as a direct replacement for softmax attention with no additional training, it can also accelerate pre-training or fine-tuning as well. Although `MonarchAttention` cannot be applied at *inference* time for autoregressive Transformers, it can be applied to *training* such models. However it is not obvious how to efficiently incorporate the causal mask into `MonarchAttention`.

Beyond Monarch & Softmax. Finally, although `MonarchAttention` is specifically designed for approximating softmax attention via Monarch, this idea can be generalized to other (possibly dynamic) structured matrix classes, as well as other nonlinear mappings. Monarch, while efficient, allocates the same number of parameters to each uniformly sized block. Yet, in practice, different regions of the attention matrix may require more fine-grained approximation. On the other hand, α-entmax mappings (Peters et al., 2019), which generalize softmax, can produce sparser attention matrices, thereby resulting in better approximations via block rank-one type matrices.

Acknowledgement

LB and CY were supported in part by NSF CAREER award CCF-1845076, NSF award CCF-2331590, and an Intel Early Career award. LB was also supported by the University of Michigan Crosby award. CY and AX were supported by NSF CCF 312842. PA and CL were supported in part by COGNISENSE, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. We thank Samet Oymak (University of Michigan) for discussion and use of computational resources provided by an Amazon Research Award on Foundation Model Development.

References

Mathieu Blondel, Andre Martins, and Vlad Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 606–615. PMLR, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34: 17413–17426, 2021.

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Ré. Pixelated butterfly: Simple and efficient sparse training for neural network models. In *International Conference on Learning Representations*, 2022.

Yuejie Chi, Yue M Lu, and Yuxin Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *IEEE Transactions on Signal Processing*, 67(20):5239–5269, 2019.

Theodore S Chihara. *An Introduction to Orthogonal Polynomials*. Courier Corporation, 2014.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

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

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *International conference on machine learning*, pages 1517–1527. PMLR, 2019.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pages 4690–4721. PMLR, 2022a.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022b.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G Dimakis. Smyrf-efficient attention using asymmetric clustering. *Advances in Neural Information Processing Systems*, 33:6476–6489, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Dan Fu, Simran Arora, Jessica Grogan, Isys Johnson, Evan Sabri Eyuboglu, Armin Thomas, Benjamin Spector, Michael Poli, Atri Rudra, and Christopher Ré. Monarch mixer: A simple sub-quadratic gemm-based architecture. *Advances in Neural Information Processing Systems*, 36:77546–77603, 2023.

Insu Han, R Jayaram, A Karbasi, V Mirrokno, D Woodruff, and A Zandieh. Hyperattention: Long-context attention in near-linear time. In *International Conference on Learning Representations*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Thomas Kailath, Sun-Yuan Kung, and Martin Morf. Displacement ranks of matrices and linear equations. *Journal of Mathematical Analysis and Applications*, 68(2):395–407, 1979.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, 2014.

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

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, 2022.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, 2021.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Victor Pan. *Structured matrices and polynomials: unified superfast algorithms*. Springer Science & Business Media, 2001.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

Hongbin Pei, Bingzhe Wei, Kevin Chen Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Ben Peters, Vlad Niculae, and André FT Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, 2019.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*, 2022.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

Ladislav Rampášek, Michael 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*, 35:14501–14515, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

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

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14138–14148, 2021.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *International Conference on Learning Representations*, 2024.

Appendix

A Equivalence of Softmax Definitions

Consider the optimization problem in (2):

$$\max_{\boldsymbol{a}} \quad f(\boldsymbol{a}) := \sum_i \boldsymbol{a}_i \boldsymbol{z}_i - \sum_i \boldsymbol{a}_i \log \boldsymbol{a}_i \quad \text{s.t.} \quad \boldsymbol{a}_i \geq 0, \quad \sum_i \boldsymbol{a}_i = 1.$$

From the KKT stationarity condition, we have

$$\frac{\partial}{\partial \boldsymbol{a}_i}\left(f(\boldsymbol{a}) + \lambda\left(1 - \sum_i \boldsymbol{a}_i\right) + \sum_i \boldsymbol{\mu}_i \boldsymbol{a}_i\right) = 0 \implies \boldsymbol{z}_i - (1 + \log \boldsymbol{a}_i) - \lambda + \boldsymbol{\mu}_i = 0,$$

where $\lambda \in \mathbb{R}, \boldsymbol{\mu} \in \mathbb{R}^N$ are dual variables. From complementary slackness $\boldsymbol{a}_i \boldsymbol{\mu}_i = 0$ and the fact that $\log \boldsymbol{a}_i$ is not defined for $\boldsymbol{a}_i = 0$, we must have $\mu_i = 0$, which gives

$$\log \boldsymbol{a}_i = \boldsymbol{z}_i - \lambda - 1 \implies \boldsymbol{a}_i = \exp(\boldsymbol{z}_i)/\exp(\lambda + 1).$$

Finally, from the constraint $\sum_i \boldsymbol{a}_i = 1$, we must have $\exp(\lambda + 1) = \sum_j \exp(\boldsymbol{z}_j)$, which gives the form of softmax in (1).

B Monarch Background

We provide an example of the transpose permutation $\boldsymbol{P}$ in Section 2. Recall that applying $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ to a vector $\boldsymbol{x} \in \mathbb{R}^N$ corresponds to row-major reshaping $\boldsymbol{x}$ to $\mathbb{R}^{m \times b}$, transposing to $\mathbb{R}^{b \times m}$, then row-major flattening back to $\mathbb{R}^N$. This is equivalent to applying a permutation matrix whose $(i+1)$-th row is given by $\boldsymbol{e}_{\sigma(i)+1}$ where

$$\sigma(i) = b \cdot (i \bmod m) + \left\lfloor \frac{i}{m} \right\rfloor, \quad i \in \{0, \ldots, N-1\}.$$

As an illustrative example, let $N = 6$, $b = 3$, and $m = 2$. The action of $\boldsymbol{P}$ is given by the following steps:

$$\begin{bmatrix}1\\2\\3\\4\\5\\6\end{bmatrix} \xrightarrow{\text{reshape } 2\times 3} \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} \xrightarrow{\text{transpose}} \begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix} \xrightarrow{\text{flatten}} \begin{bmatrix}1\\4\\2\\5\\3\\6\end{bmatrix}.$$

In matrix form, we have

$$\boldsymbol{P} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 1 & 0 & 0\\0 & 1 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 1 & 0\\0 & 0 & 1 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}.$$

C Details for `MonarchAttention`

C.1 Updates

Derivatives. We evaluate f with $\boldsymbol{A} = \boldsymbol{M}$ as a Monarch matrix. Using (5), we have

$$\begin{aligned} f(\boldsymbol{M}; \boldsymbol{Q}, \boldsymbol{K}) &= \sum \boldsymbol{L}_{jkl} \boldsymbol{R}_{kji} \overline{\boldsymbol{Q}}_{jlv} \overline{\boldsymbol{K}}_{kiv} - \sum \boldsymbol{L}_{jkl} \boldsymbol{R}_{kji} \log(\boldsymbol{L}_{jkl} \boldsymbol{R}_{kji}) \\ &= \sum \boldsymbol{L}_{jkl} \boldsymbol{R}_{kji} \overline{\boldsymbol{Q}}_{jlv} \overline{\boldsymbol{K}}_{kiv} - \sum \boldsymbol{L}_{jkl} \boldsymbol{R}_{kji} \log \boldsymbol{R}_{kji} - \sum \boldsymbol{R}_{kji} \boldsymbol{L}_{jkl} \log \boldsymbol{L}_{jkl}. \end{aligned}$$

The derivatives of f w.r.t. each factor are given by

$$\frac{\partial f(\boldsymbol{M};\boldsymbol{Q},\boldsymbol{K})}{\partial \boldsymbol{L}_{jkl}} = \boldsymbol{\beta}_{L,jkl} - \boldsymbol{c}_{L,jk} - (1+\log \boldsymbol{L}_{jkl})\boldsymbol{\gamma}_{L,jk}, \tag{11}$$

$$\frac{\partial f(\boldsymbol{M};\boldsymbol{Q},\boldsymbol{K})}{\partial \boldsymbol{R}_{kji}} = \boldsymbol{\beta}_{R,kji} - \boldsymbol{\gamma}_{R,kj} - (1+\log \boldsymbol{R}_{kji})\boldsymbol{c}_{R,kj}, \tag{12}$$

where

$$\boldsymbol{\beta}_{L,jkl} = \sum_v \overline{\boldsymbol{Q}}_{jlv} \sum_i (\boldsymbol{R}_{kji}\overline{\boldsymbol{K}}_{kiv}), \quad \boldsymbol{c}_{L,jk} = \sum_i \boldsymbol{R}_{kji}\log \boldsymbol{R}_{kji}, \quad \boldsymbol{\gamma}_{L,jk} = \sum_i \boldsymbol{R}_{kji},$$

$$\boldsymbol{\beta}_{R,kji} = \sum_v \overline{\boldsymbol{K}}_{kiv} \sum_l (\boldsymbol{L}_{jkl}\overline{\boldsymbol{Q}}_{jlv}), \quad \boldsymbol{\gamma}_{R,kj} = \sum_l \boldsymbol{L}_{jkl}\log \boldsymbol{L}_{jkl}, \quad \boldsymbol{c}_{R,kj} = \sum_l \boldsymbol{L}_{jkl}.$$

We derive updates for each factor based on maximizing f with the other factor fixed.

***L* update.** First, we fix $\boldsymbol{R} \in \Delta^{m\times b\times b}$ and consider

$$\max_{\boldsymbol{L}} \quad f(\boldsymbol{M};\boldsymbol{Q},\boldsymbol{K}) \quad \text{s.t.} \quad \boldsymbol{L}_{jkl} \geq 0, \quad \sum_k \boldsymbol{L}_{jkl} = 1.$$

From the KKT stationarity condition, we have

$$\frac{\partial}{\partial \boldsymbol{L}_{jkl}}\left(f(\boldsymbol{M};\boldsymbol{Q},\boldsymbol{K}) + \sum \boldsymbol{\lambda}_{L,jl}\left(1 - \sum_k \boldsymbol{L}_{jkl}\right) + \sum \boldsymbol{\mu}_{L,jkl}\boldsymbol{L}_{jkl}\right) = 0$$

where $\boldsymbol{\lambda}_L \in \mathbb{R}^{b\times m}$, $\boldsymbol{\mu}_L \in \mathbb{R}^{b\times m\times m}$ are dual variables. Along with (11), we have

$$\boldsymbol{\beta}_{L,jkl} - \boldsymbol{c}_{L,jk} - (1+\log \boldsymbol{L}_{jkl})\boldsymbol{\gamma}_{L,jk} - \boldsymbol{\lambda}_{L,jl} + \boldsymbol{\mu}_{L,jkl} = 0.$$

Now, from complementary slackness $\boldsymbol{\mu}_{L,jkl}\boldsymbol{L}_{jkl} = 0$ and the fact that $\log \boldsymbol{L}_{jkl}$ is not defined for $\boldsymbol{L}_{jkl} = 0$, we must have $\boldsymbol{\mu}_{L,jkl} = 0$. Moreover, since $\boldsymbol{R} \in \Delta^{m\times b\times b}$, we have $\boldsymbol{\gamma}_{L,jk} = 1$. Altogether, we have

$$\log \boldsymbol{L}_{jkl} = \boldsymbol{\beta}_{L,jkl} - \boldsymbol{c}_{L,jk} - \boldsymbol{\lambda}_{L,jl} - 1 \implies \boldsymbol{L}_{jkl} = \frac{\exp(\boldsymbol{\beta}_{L,jkl} - \boldsymbol{c}_{L,jk})}{\exp(\boldsymbol{\lambda}_{L,jl} + 1)}.$$

Finally, from the constraint $\sum_k \boldsymbol{L}_{jkl} = 1$, we must have $\exp(\boldsymbol{\lambda}_{L,jl}+1) = \sum_k \exp(\boldsymbol{\beta}_{L,jkl} - \boldsymbol{c}_{L,jk})$, which gives the final closed form update:

$$\boldsymbol{L} = \mathrm{softmax}_k(\boldsymbol{Z}_L), \quad \boldsymbol{Z}_{L,jkl} = \boldsymbol{\beta}_{L,jkl} - \boldsymbol{c}_{L,jk},$$

where $\mathrm{softmax}_k$ is applied along the k index dimension.

***R* update.** Similarly, we fix $\boldsymbol{L} \in \Delta^{b\times m\times m}$ and consider

$$\max_{\boldsymbol{R}} \quad f(\boldsymbol{M};\boldsymbol{Q},\boldsymbol{K}) \quad \text{s.t.} \quad \boldsymbol{R}_{kji} \geq 0, \quad \sum_i \boldsymbol{R}_{kji} = 1.$$

From the KKT stationarity condition, we have

$$\frac{\partial}{\partial \boldsymbol{R}_{kji}}\left(f(\boldsymbol{M};\boldsymbol{Q},\boldsymbol{K}) + \sum \boldsymbol{\lambda}_{R,kj}\left(1 - \sum_i \boldsymbol{R}_{kji}\right) + \sum \boldsymbol{\mu}_{R,kji}\boldsymbol{R}_{kji}\right) = 0$$

where $\boldsymbol{\lambda}_R \in \mathbb{R}^{m\times b}$, $\boldsymbol{\mu}_R \in \mathbb{R}^{m\times b\times b}$ are dual variables. Along with (12), we have

$$\boldsymbol{\beta}_{R,kji} - \boldsymbol{\gamma}_{R,kj} - (1+\log \boldsymbol{R}_{kji})\boldsymbol{c}_{R,kj} - \boldsymbol{\lambda}_{R,kj} + \boldsymbol{\mu}_{R,kji} = 0.$$

As before, from complementary slackness $\boldsymbol{\mu}_{R,kji}\boldsymbol{R}_{kji} = 0$ and the fact that $\log \boldsymbol{R}_{kji}$ is not defined for $\boldsymbol{R}_{kji} = 0$, we must have $\boldsymbol{\mu}_{R,kji} = 0$. Altogether, we have

$$\log \boldsymbol{R}_{kji} = \frac{\boldsymbol{\beta}_{R,kji} - \boldsymbol{\gamma}_{R,kj} - \boldsymbol{\lambda}_{R,kj}}{\boldsymbol{c}_{R,kj}} - 1 \implies \boldsymbol{R}_{kji} = \frac{\exp(\boldsymbol{\beta}_{R,kji}/\boldsymbol{c}_{R,kj})}{\exp(\boldsymbol{\gamma}_{R,kj}/\boldsymbol{c}_{R,kj} + \boldsymbol{\lambda}_{R,kj}/\boldsymbol{c}_{R,kj} + 1)}.$$

Finally, from the constraint $\sum_i \boldsymbol{R}_{kji} = 1$, we must have $\exp(\boldsymbol{\gamma}_{R,kj}/\boldsymbol{c}_{R,kj} + \boldsymbol{\lambda}_{R,kj}/\boldsymbol{c}_{R,kj} + 1) = \sum_i \exp(\boldsymbol{\beta}_{R,kji}/\boldsymbol{c}_{R,kj})$, which gives the final closed form update:

$$\boldsymbol{R} = \mathrm{softmax}_i(\boldsymbol{Z}_R), \quad \boldsymbol{Z}_{R,kji} = \boldsymbol{\beta}_{R,kji}/\boldsymbol{c}_{R,kj},$$

where $\mathrm{softmax}_i$ is applied along the i index dimension.

C.2 Naïve Algorithm

We provide pseudo-code for the naive version of `MonarchAttention` in Figure 7. This is a direct implementation of alternating maximization for finding the Monarch factors, where we highlight the choices to (1) initialize $\boldsymbol{L}$ to be block identity, and (2) update $\boldsymbol{R}$ before $\boldsymbol{L}$ in each iteration. We include this here for completeness, so that readers can directly compare this code with the algorithmic steps. The implementation in Figure 4 is a reorganization of Figure 7 where $\boldsymbol{L}, \boldsymbol{R}$ are not materialized in HBM, resulting in `FlashAttention`-like kernels.

```
# Q: array of size (N, d)
# K: array of size (N, d)
# V: array of size (N, d)
# T: number of steps

def monarch_attention(Q, K, V, T):
    L = stack(b * [eye(m)])
    Qb = einshape("(lj)v->jlv", Q, j=b)
    Kb = einshape("(ki)v->kiv", K, i=b)

    # Alternating maximization for L, R
    for t in range(T):
        # R update
        aR = einsum("jkl,jlv->kjv", L, Qb)
        bR = einsum("kjv,kiv->kji", aR, Kb)
        cR = einsum("jkl->kj", L)
        R = softmax(bR / cR[:, :, None], axis=2)

        # L update
        aL = einsum("kji,kiv->jkv", R, Kb)
        bL = einsum("jkv,jlv->jkl", aL, Qb)
        cL = einsum("kji->jk", R * log(R))
        L = softmax(bL - cL[:, :, None], axis=1)

    # Monarch multiply
    Vb = einshape("(ki)v->kiv", V, i=b)
    Y = einsum("kji,kiv->jkv", R, Vb)
    Z = einsum("jkl,jkv->ljv", L, Y)
    O = einshape("ljv->(lj)v", Z)

    return O
```

Figure 7: Naïve implemention of `MonarchAttention`

C.3 Padding and Masking

In practice, the sequence length N may not be divisible by the desired block size b. In such cases, we round the number of blocks m to $m' = \lceil N/b \rceil$, and set the new sequence length $N' = m'b$, post-padding $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ to have N' rows. However, we need to take special care that the final $N' - N$ columns of the padded Monarch attention matrix $\boldsymbol{M} \in \Delta^{N' \times N'}$ are zero, since these correspond to padded rows of $\boldsymbol{V}$. This is also an issue when batched sequences of different lengths are padded to a maximum length to avoid dynamic resizing.

From (5), it is clear that to set all columns of $\boldsymbol{M}$ beyond the N-th column to zero, it is sufficient to set $\boldsymbol{R}_{kji}$ to zero whenever $b(k-1) + i > N$. Thus, we simply form the mask $\boldsymbol{\omega} \in \mathbb{R}^{m' \times b \times b}$ given by

$$\boldsymbol{\omega}_{kji} = \begin{cases} 0 & b(k-1) + i \leq N \\ -\infty & \text{otherwise,} \end{cases}$$

which we then add to $\boldsymbol{Z}_R$ before softmax in (7). We can also pre-pad the sequence, which would be change the above condition to $b(k-1) + i \geq N' - N$.

D Experimental Details

D.1 Baselines

We describe the baselines used in Section 4.

- `linear-attention` (Katharopoulos et al., 2020) approximates $\exp(\boldsymbol{q}^\top \boldsymbol{k}) \approx \phi(\boldsymbol{q})^\top \phi(\boldsymbol{k})$ where $\phi : \mathbb{R}^d \to \mathbb{R}^r$ is a kernel feature map, resulting in a rank r approximation to softmax attention:
$$\operatorname{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top) \approx \frac{\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\top}{\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\top \mathbf{1}_N \mathbf{1}_N^\top}.$$
Katharopoulos et al. (2020) propose the map $\phi(\boldsymbol{x}) = 1 + \operatorname{elu}(\boldsymbol{x})$ with $r = d$ where elu is the exponential linear unit applied element-wise.
- `performer` (Choromanski et al., 2021) is a linear attention method using the fact that
$$\exp(\boldsymbol{q}^\top \boldsymbol{k}) = \mathbb{E}_{\boldsymbol{\omega}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}_d)}\left[\exp\left(\boldsymbol{\omega}^\top \boldsymbol{q} - \frac{\|\boldsymbol{q}\|^2}{2}\right)\exp\left(\boldsymbol{\omega}^\top \boldsymbol{k} - \frac{\|\boldsymbol{k}\|^2}{2}\right)\right]$$
to construct a random kernel feature map
$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{r}}\exp\left(-\frac{\|\boldsymbol{x}\|^2}{2}\right)\begin{bmatrix}\exp(\boldsymbol{\omega}_1^\top \boldsymbol{x}) & \dots & \exp(\boldsymbol{\omega}_r^\top \boldsymbol{x})\end{bmatrix}^\top,$$
where $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_r \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$.
- `cosformer` (Qin et al., 2022) is a linear attention method utilizing position-dependent kernel feature maps of the form
$$\phi_i(\boldsymbol{x}) = \begin{bmatrix}\sin\left(\frac{\pi i}{2N}\right)\operatorname{relu}(\boldsymbol{x}_i) & \cos\left(\frac{\pi i}{2N}\right)\operatorname{relu}(\boldsymbol{x}_i)\end{bmatrix}, \ \forall i \in [N],$$
which produces a rank $r = 2d$ approximation.
- `nystromformer` (Xiong et al., 2021) computes landmark $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{K}} \in \mathbb{R}^{r\times d}$ from $\boldsymbol{Q}, \boldsymbol{K}$ by averaging N/r consecutive spans of rows, which are used to approximate softmax attention via the quadrature method:
$$\tilde{\boldsymbol{F}} = \operatorname{softmax}(\boldsymbol{Q}\tilde{\boldsymbol{K}}^\top),\ \tilde{\boldsymbol{B}} = \operatorname{softmax}(\tilde{\boldsymbol{Q}}\boldsymbol{K}^\top),\ \tilde{\boldsymbol{A}} = \operatorname{softmax}(\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{K}}^\top)$$
$$\operatorname{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top) \approx \tilde{\boldsymbol{F}}\tilde{\boldsymbol{A}}^+\tilde{\boldsymbol{B}},$$
where $\tilde{\boldsymbol{A}}^+$ denotes the pseudoinverse of $\tilde{\boldsymbol{A}}$, producing a rank r approximation.

D.2 Image Classification with ViT

The ViT-B model fine-tuned on ImageNet-21K is retrieved from the Hugging Face `transformers` library (Wolf et al., 2019) as `google/vit-base-patch16-224`. The ImageNet-1K evaluation dataset is retrieved from the Hugging Face `datasets` library (Lhoest et al., 2021) as `imagenet-1k` using the `validation` split. We vary the following hyperparameters:

- `monarch-attention`: $b = 14$ and $T \in \{1, 2, 3\}$
- `performer`: $r \in \{16, 32, 48, 64, 80, 96\}$
- `nystromformer`: $r \in \{16, 24, 32, 40\}$

The choice of $b = 14$ for `monarch-attention` is due to the fact that images are patched in a 14×14 grid. We also apply post-padding as described in Appendix C.3 for the `[CLS]` token, since it is appended at the end of the sequence.

D.3 Question Answering with RoBERTa

The RoBERTa-B model fine-tuned on SQuAD1.1 is retrieved from the Hugging Face `transformers` library as `csarron/roberta-base-squad-v1`. The SQuAD1.1 evaluation dataset is retrieved from the Hugging Face `datasets` library as `squad` using the `validation` split. For evaluation, we truncate and pad to sequence length of 384. We vary the following hyperparameters:

Table 2: Hyperparameters used for BART summarization.

Sequence Length	Method	(b, T)	r	Total Attention FLOPs (10^9)
1024	Softmax	–	–	9.66
	Nyströmformer	–	64	1.93
	`MonarchAttention`	(32, 3)	–	1.96
2048	Softmax	–	–	38.7
	Nyströmformer	–	80	4.41
	`MonarchAttention`	(32, 2)	–	3.93
4096	Softmax	–	–	155.
	Nyströmformer	–	112	10.6
	`MonarchAttention`	(64, 2)	–	10.9
8192	Softmax	–	–	619.
	Nyströmformer	–	160	35.0
	`MonarchAttention`	(64, 2)	–	31.4

- `monarch-attention`: $T = 1$ and $b \in \{24, 48, 96, 128\}$
- `performer`: $r \in \{32, 64, 96, 128, 160, 192\}$
- `nystromformer`: $r \in \{16, 32, 48, 64\}$

D.4 Summarization with BART

The pre-trained BART-B model is retrieved from the Hugging Face `transformers` library as `facebook/bart-base`. The BookSum-chapters training/evaluation dataset is retrieved from the Hugging Face `datasets` library as `kmfoda/booksum` using the `train` and `validation` splits respectively. BART employs learned positional embeddings up to 1024 sequence length, and since we are interested in long-sequence summarization up to 8192 tokens, we linearly interpolate the encoder positional embeddings up to 8192 tokens, before fine-tuning on BookSum-chapters – we leave the decoder positional embeddings intact. We fine-tune for 5 epochs with batch size of 32 and learning rate of 10^{-4} using the Adam optimizer (Kingma and Ba, 2014) without weight decay, with the input and summary sequences truncated and padded to 8192 and 512 tokens respectively. For evaluation, we truncate the input sequence to the corresponding sequence length in Figure 3. The hyperparameters for each method across sequence lengths are shown in Table 2.

D.5 Image Generation with DiT

The pre-trained model DiT-XL is retrieved from the Hugging Face `transformers` library as `facebook/DiT-XL-2-256`. Following Peebles and Xie (2023), we generate images using 32 sampling steps, a 2×2 patch size, and a classifier-free guidance scale of 1.5. We use the following hyperparameters:

- `monarch-attention`: $b = 16$ and $T = 3$
- `nystromformer`: $r = 32$

To create the images in Figure 5, we used a random seed of 0 input the *same* 36 random Gaussian samples into all three models. To obtain the results in Table 1, we again used a random seed of 0, and generated $50K$ images from each type of model, again using the *same* $50K$ random samples across all models.

E Additional Experimental Results

E.1 Convergence of `MonarchAttention`

We empirically show that `MonarchAttention` converges very quickly in the ViT image classification setting described in Appendix D.2. We measure of the value of the softmax variational objective

$f(\boldsymbol{M}^{(T)}; \boldsymbol{Q}, \boldsymbol{K})$ as defined in (6) in the 0-th layer and 5-th head for the first few T – the results are shown in Table 3. We see that by $T = 2$, the variational objective has already reached a stationary point, with even $T = 1$ closing most of the gap. This demonstrates how `MonarchAttention` is able to reach an accurate solution with even just one step.

Table 3: Convergence of $f(\boldsymbol{M}^{(T)}; \boldsymbol{Q}, \boldsymbol{K})$.

T	0	1	2	3
$f(\boldsymbol{M}^{(T)})$	418	937	940	940

E.2 Layer Ablation for RoBERTa Question Answering

We investigate the impact of replacing different layers with `MonarchAttention` in the RoBERTa question answering setting described in Appendix D.3. The results are shown in Table 4.

Table 4: F1 score when replacing a given layer of RoBERTa with `MonarchAttention`.

Layer	0	1	2	3	4	5	6	7	8	9	10	11
F1 (↑)	92.7	92.5	92.6	92.0	91.4	91.9	86.3	91.1	92.7	93.2	93.0	92.3

The initial and final layers are more amenable to replacement compared to the middle layers.

E.3 Post-Swap Fine-Tuning for BART Summarization

We demonstrate the improved performance of `MonarchAttention` when fine-tuning on BART summarization post-replacement. We follow the setting and hyperparameters of Appendix D.4, but fine-tune with `MonarchAttention` or Nyströmformer instead of the standard softmax attention for sequence length $N = 8192$. The results are shown in Table 5, where we also copy the zero-shot results from the main body for reference.

Table 5: Post-swap fine-tuning vs zero-shot performance on BART summarization.

Type	**Method**	**Total Attention FLOPs** (10^9)	**ROUGE-1** (↑)
–	Softmax	618	35.57
Fine-tuned	`MonarchAttention`	31	**35.38**
	Nyströmformer	35	33.94
Zero-shot	`MonarchAttention`	31	**34.60**
	Nyströmformer	35	32.86

We see that `MonarchAttention` is stable under fine-tuning, and is indeed a strong approximation even beyond the zero-shot setting. In particular, we can almost fully close the gap with `MonarchAttention` to the original softmax model, while benefiting from more efficient fine-tuning with roughly 5% of the attention FLOPs.

E.4 Layer Ablation for DiT Image Generation

We investigate the impact of replacing different quarters with `MonarchAttention` in the DiT image generation setting described in Appendix D.5. The results are shown in Table 6. We see that replacing earlier layers is more favorable than later layers, with the most degradation resulting from replacing the final layers of the DiT.

E.5 Graph Node Classification with GraphGPS

We evaluate the efficacy of `MonarchAttention` beyond language and image modalities by considering zero-shot conversion of attention in the GPS graph Transformer (GPS-GT) (Rampášek et al.,

Table 6: (s)FID scores when replacing different quarters of DiT with `MonarchAttention`.

Quarter	First	Second	Third	Fourth
FID (↓)	0.07	0.21	0.34	1.35
sFID (↓)	0.14	0.39	0.52	2.23

2022) on the Actor graph dataset (Pei et al., 2020) for node classification. In particular, we convert the attention layer in a single-layer 170K parameter GPS-GT with 2 heads, sequence length $N = 7600$, and head dimension $d = 48$. We train for 1500 steps with learning rate of 5×10^{-4} using the Adam optimizer, and also evaluate on the train split[4]. To evaluate the performance at different FLOP counts, we vary the number of steps $T \in \{1, 2, 3, 4\}$ while fixing $b = 128$ for `MonarchAttention`, and vary the rank $r \in \{128, 192, 256, 320\}$ for Nyströmformer. The results are shown in Figure 8.

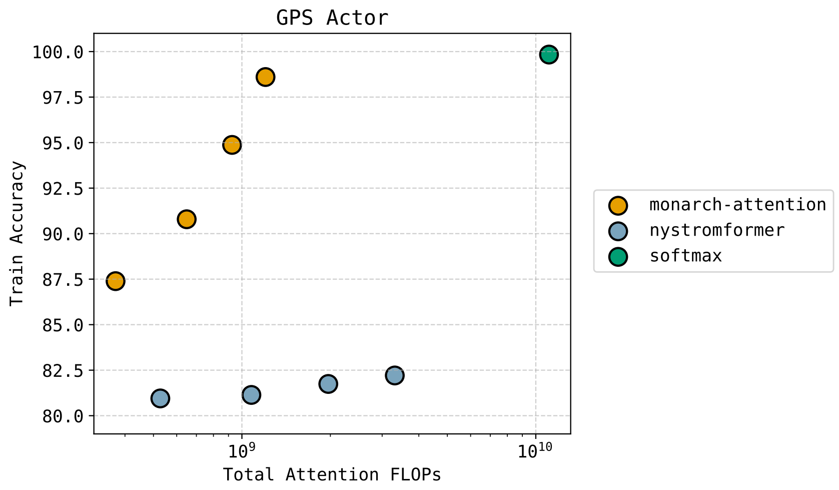


Figure 8: **Zero-shot conversion of attention layers for graph node classification.** We vary hyperparameters to evaluate training accuracy vs. total attention FLOPs across all layers for GPS on the Actor dataset.

`MonarchAttention` achieves a favorable quality-efficiency tradeoff, almost matching the softmax model quality with an order of magnitude less FLOPs. On the other hand, Nyströmformer achieves roughly 18% less accuracy with 2-3× more FLOPs than `MonarchAttention`.

[4]We evaluate on the train split due to low performance on the validation/test sets due to limited data/training from scratch. Since we are primarily interested in the post-train zero-shot approximation capabilities of `MonarchAttention`, this does not introduce bias in the evaluation.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We support the claims made made in the abstract and introduction through extensive experiments in Section 4.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss several limitations of the proposed algorithm in Section 5, whose addressing we pose as future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [NA]

Justification: No theoretical results in this paper.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide exact Python-like pseudocode for the proposed algorithm, along with precise set-ups for each experiment.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will release code for all experiments after the review process. We also specify publicly-available model checkpoints used for each experiment.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide full details for hyperparameters and settings used in Section 4 and the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: There is little to no randomness in the experiments carried out in this paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We specify the GPU used for benchmarking run-times in Section 4.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: All research conducted conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: We believe this work to be foundational and not tied to any specific application or societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: We do not release any new data or models.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite and provide publicly available links to all data and models used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: No new assets are developed.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.