Real-time Core-Periphery Guided ViT with Smart Data Layout Selection on Mobile Devices

Zhihao Shu[1*] **Xiaowei Yu**[2*] **Zihao Wu**[1] **Wenqi Jia**[2] **Yinchen Shi**[3]
Miao Yin[2] **Tianming Liu**[1] **Dajiang Zhu**[2] **Wei Niu**[1]
[1]University of Georgia [2]University of Texas at Arlington [3]NYU
{Zhihao.Shu, Zihao.Wu1, wniu}@uga.edu
tliu@cs.uga.edu
ys4653@nyu.edu
{xxy1302, wxj1489}@mavs.uta.edu
{dajiang.zhu, miao.yin}@uta.edu

Abstract

Mobile devices have become essential enablers for AI applications, particularly in scenarios that require real-time performance. Vision Transformer (ViT) has become a fundamental cornerstone in this regard due to its high accuracy. Recent efforts have been dedicated to developing various transformer architectures that offer improved accuracy while reducing the computational requirements. However, existing research primarily focuses on reducing the theoretical computational complexity through methods such as local attention and model pruning, rather than considering realistic performance on mobile hardware. Although these optimizations reduce computational demands, they either introduce additional overheads related to data transformation (e.g., Reshape and Transpose) or irregular computation/data-access patterns. These result in significant overhead on mobile devices due to their limited bandwidth, which even makes the latency worse than vanilla ViT on mobile. In this paper, we present ECP-ViT, a real-time framework that employs the core-periphery principle inspired by the brain functional networks to guide self-attention in ViTs and enable the deployment of ViT models on smartphones. We identify the main bottleneck in transformer structures caused by data transformation and propose a hardware-friendly core-periphery guided self-attention to decrease computation demands. Additionally, we design the system optimizations for intensive data transformation in pruned models. ECP-ViT, with the proposed algorithm-system co-optimizations, achieves a speedup of $4.6\times$ to $26.9\times$ on mobile GPUs across four datasets: STL-10, CIFAR100, TinyImageNet, and ImageNet.

1 Introduction

In recent decades, there has been a significant increase in applying deep neural network (DNN) architectures across various fields, including autonomous driving [23], natural language processing [9], extended reality (XR) [13], image processing [22], and View Synthesis [32]. Along with significant progress in hardware performance and public datasets, more and more complex DNN architectures have been proposed, including Convolution Neural Network [52], RNN [36], Transformer [51]. These architectures have led to groundbreaking breakthroughs in various application domains by leveraging their powerful feature extraction abilities. Meanwhile, mobile devices have become essential for deploying these applications, especially in scenarios that demand real-time performance. The mobile GPU on Oneplus 11 can achieve a theoretical *peak performance of over 3T FLOPS* (floating point

[*]Equal contributions.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3034

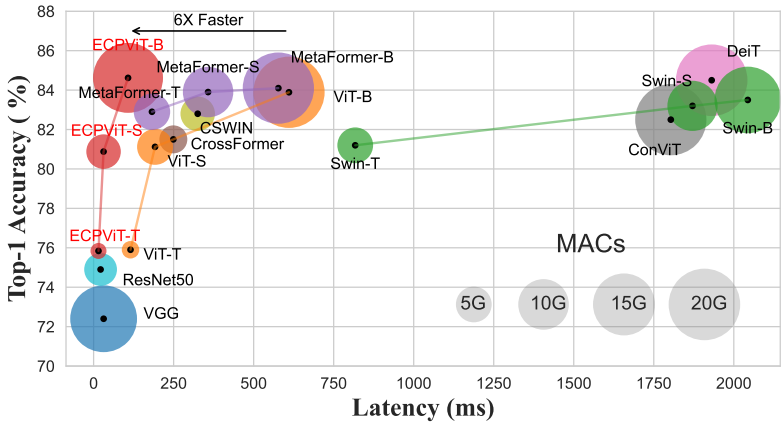

Figure 1: Accuracy-Latency-MACs comparison among various models on ImageNet-1K. Latency is measured on Oneplus 11 (Snapdragon 8 Gen 2 SoC). The radius of each circle represents the model's MACs (Multiply-Accumulate Operations). Our ECP-ViT-Base achieves the highest top-1 accuracy and offers the fastest inference speed among transformer-based models, providing 6× faster than ViT-Base. We simplify the notation as "Base" = B, "Small" = S, and "Tiny" = T.

operations per second) [1]. Their widespread availability and increasing computational capabilities make them ideal platforms for extending the impact of these algorithm innovations.

Compared to traditional convolutional neural networks (CNNs), transformers have recently become widely used across various areas and tasks due to their high accuracy. The attention mechanism in the transformer allows neurons to exchange messages effectively and efficiently, leading to promising results in natural language processing [48, 16] and computer vision domains [17, 53]. However, transformer architectures also known for their deeper network layers and require frequent reshaping and transposing of the feature maps. This results in more intermediate results and leads to a memory bound for the computation. Optimizing transformers for efficient execution becomes particularly challenging in environments where memory bandwidth is limited.

Advancements in transformer architecture design, including Vision Transformer (ViT) [17], have focused on improving message exchange mechanisms among spatial tokens through different Token Mixers. Other efforts include the shifted window attention in Swin [34], the token-mixing MLP in Mixer [44], and the pooling in MetaFormer [57]. These designs aim to enhance self-attention accuracy and hardware efficiency compared to the original vanilla ViT [17], enabling more effective and efficient execution on various hardware platforms. Despite the significant progress made in transformer architecture, particularly in reducing theoretical computation demands, achieving real-time performance for transformer-based models on mobile devices remains a major challenge. Figure 1 illustrates the latency, accuracy, and floating-point operations (FLOPs) comparison of different models on a mobile GPU. Despite having fewer FLOPs, Swin-T (local attention) is even more than 3× slower than ViT-T (global attention). This is because local attention requires more frequent data transformation to reorganize tokens, which accounts for 69% of execution time in Swin. In contrast, these numbers are 43% in ViT and only 0.8% in VGG16. Similarly, DeiT has a broadly similar structure to ViT, but it involves more complex data layout transformations and larger intermediate results, as illustrated in Table 1. Unlike powerful server platforms, mobile devices have limited memory bandwidth [24], making it challenging to benefit from reduced theoretical FLOPs. There has been a fundamental lack of general principles that involve co-designing model architecture and system optimizations.

In this work, we present an integrated framework that incorporates co-optimizations by revisiting the model design and system optimizations. First, we propose a hardware-efficient and computational-friendly sparse scheme (guided by the Core-Periphery principle in brain networks) [61, 59, 60, 58, 62] that can be applied to guide the message exchange in self-attention. This scheme also helps reduce bandwidth pressure for the subsequent Softmax operation. Secondly, we develop a set of comprehensive compiler-based optimizations **supporting the proposed pruning scheme and**

Table 1: Comparison between ViT and DeiT. All data is measured on Oneplus 11 (Snapdragon 8 Gen 2 SoC). Layout Transformation indicates the time spent on transforming the tensor's layout, such as Transpose and Reshape. Computation indicates the time spent on pure tensor computation.

Model	Layout Transformation (ms)	Computation (ms)	Intermediate Results (MB)
ViT-Base [17]	324	286	106.47
DeiT-Base [46]	1303	633	125.42

fully eliminate the overhead for Transpose and Reshape. Enabled by the advanced compiler optimizations, it is possible to achieve both high accuracy and high acceleration simultaneously. We demonstrate that, 1) our proposed fine-grained structured core-periphery guided self-attention offers advantages in both accuracy and speed, and 2) our compiler framework exhibits superior end-to-end acceleration performance for both the original and proposed ECP-ViT models.

We summarize our contributions in three aspects:

- We incorporate the organizational principle of brain functional networks—specifically, the core-periphery principle—to guide self-attentions in ViTs. Additionally, we introduce a compiler code generation framework that efficiently supports the core-periphery structures on mobile devices.
- We develop comprehensive compiler-based optimizations that can fully eliminate the overhead of data transformation, bridging the gap between accuracy and latency.
- We build ECP-ViT, an end-to-end framework that combines algorithm and system design to achieve real-time performance on mobile devices. ECP-ViT achieves the significant speedup on off-the-shelf mobile devices, reaching up to 16.1$\times$ on ImageNet inference while maintaining an accuracy of 84.51%.

2 Background and Motivation

Mixed blessing of attention. ViTs are widely utilized as robust backbones across various tasks. The two primary types of attention are Global Attention [3, 33] and Local Attention [39, 41, 6, 42, 7]. Global Attention, as seen in standard transformer models, allows each token to attend to every other token in the input sequence. This mechanism ensures comprehensive context capture, leading to high accuracy. However, the downside is its computational intensity, as it scales quadratically with the input length, making it less efficient for longer sequences or resource-constrained platforms.

Local Attention [64, 49] restricts the focus of each token to a subset of adjacent tokens (or with specific patterns), resulting in the reduction of computational complexity. However, local Attention does not always translate into proportionate realistic speedups when compared to the theoretical reduction in floating-point operations (FLOPs). This approach significantly reduces the computational complexity but at the cost of more layout transformations. These factors offset the expected gains from reduced computational complexity, particularly in environments like mobile GPUs where memory bandwidth and efficient data handling are critical, as reflected in Figure 1. The actual performance gains need to be evaluated in the context of specific hardware and application requirements. Compared to the above two types of attention, ECP-ViT is the first work to explore a speed-aware end-to-end framework that achieves real-time performance on real-world mobile devices for ViTs.

Efficient network design and architecture search. In this category, a significant methodology is Neural Architecture Search (NAS) [67, 40, 18, 31]. Some of the work leverages network pruning and sparse training to further reduce the theoretical FLOPs. At the token level, Tang et al. [43] introduces a patch slimming method to remove redundant tokens. Evo-ViT [54] updates selected informative and uninformative tokens through distinct computation paths, while VTP [66] reduced embedding dimensionality by introducing control coefficients. At the model architecture level, UP-ViTs [56] adopts a unified approach to prune channels in ViTs. SViTE [10] dynamically extracts and trains sparse sub-networks instead of training the entire model. Despite the significant progress made by these methods, both token-sampling and data-driven strategies may heavily depend on specific datasets and tasks, limiting the generalization capability of vision transformers. Additionally, *these token-level pruning or selection introduce additional operations for the Reshape and Transpose to the feature map*, leading to fewer benefits from reduced computation complexity. In contrast, ECP-ViT achieves a significant speedup on mobile platforms through two key components: 1) the more efficient

https://doi.org/10.52202/079017-3034

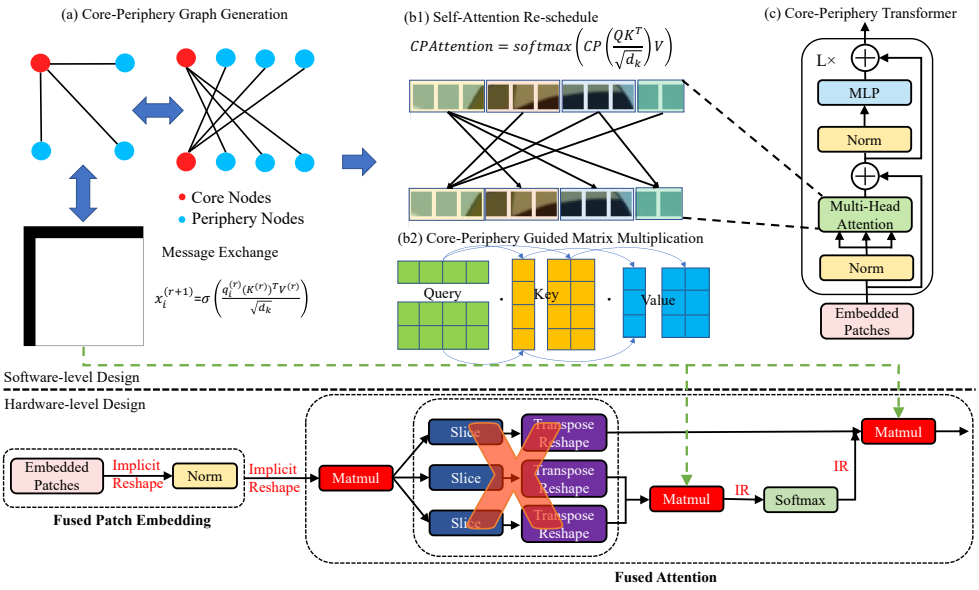

Figure 2: Co-Design of Core-Periphery Principle Guided self-attention mechanism in ECP-ViT. At the software-level design, the rescheduled interactions between patches in ECP-ViT are guided by the generated Core-Periphery (CP) graphs. Moreover, the multiplication of query, key, and value matrices in the self-attention mechanism of ECP-ViT is also under the guidance of the core-periphery graph. At the hardware-level design, the slice and transpose reshape operations are eliminated.

self-attention guided by the Core-Periphery principle, and 2) our compiler-based optimizations that eliminate data transformation overhead (i.e., `Reshape` and `Transpose`).

DNN frameworks on mobile devices. Recently, there has been a dedicated focus on developing inference acceleration frameworks for mobile devices from both academia and industry. Some efforts include MCDNN [21], DeepSense [55], MobiSR [30], and PatDNN [38]. However, none of these frameworks support the execution of transformer models on mobile devices. Other efforts have been made to optimize the execution of transformer models on mobile devices, include TFLite [2], TVM [11], MNN [26], Pytorch-Mobile [25], and DNNFusion [37]. They support optimizations including operator fusion, constant folding, and quantization on mobile devices. *However, they are not able to eliminate the intensive* `Reshape` *and* `Transpose` *operations in transformer models.* In this work, our goal is to find the most appropriate CP pruning scheme for mobile ViT acceleration and the corresponding full-stack acceleration framework.

3 Methodology

The entire framework of co-design for ECP-ViT is illustrated in Figure 2. It comprises software-level design (algorithm) and hardware-level design, both of which we will discuss in detail in the subsequent sections.

3.1 Problem Definition and Research Issues

As mentioned earlier, the attention mechanism in transformer models involves a large amount of data transformation, which poses significant challenges to hardware efficiency and deployment on mobile devices. This is because (i) *data transformation is memory-bound, requiring high memory bandwidth.* On mobile devices, which typically have restricted memory bandwidth, this leads to increased time consumption for processing these transformations; and (ii) data transformation in transformers, especially within the multi-head attention mechanism, *often results in irregular data access patterns.* This is typically observed during operations like tensor reorganization. Such irregular access patterns

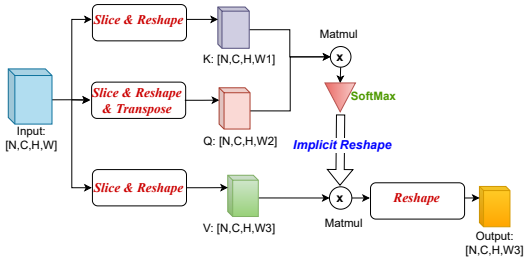

Figure 3: Illustration of ViT on attention module. In this example, we use the NCHW format for presenting our data. Words with red color are explicit data layout and transformative operators.

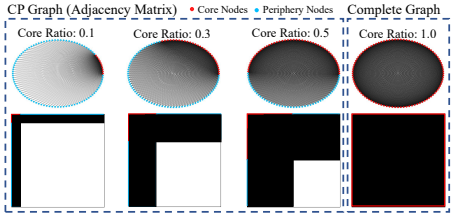

Figure 4: Examples of Core-Periphery Nodes in the graph, where the core ratio is calculated by dividing the number of core nodes by the total number of nodes. The white area represents pruned nodes. In adjacency matrices, black color indicates connections between nodes, while white represents no edge.

can significantly degrade performance, particularly in the absence of sufficient concurrent processing threads.

We classify the data transformation into two categories (as shown in Figure 3), `Explicit` and `Implicit`. `Explicit` transformations (red color operators in round boxes) are designed by the algorithm developer to ensure the model follows its intended computational logic. On the other hand, `Implicit` transformations adopt data layouts that best suit layout-sensitive operators for increased effectiveness. `Implicit` focuses more on performance optimization tailored to specific hardware platforms and depends on the inference framework. For instance, TFLite may prefer using NHWC layout for `MatMul` layer, while others may prefer NCHW ones. Based on the classification mentioned above, ECP-ViT specifically answers these three questions to achieve computation efficiency and data transformation elimination:

- How to design a *hardware-friendly pruning scheme* for ViTs without compromising accuracy?
- How to *effectively minimize the data transformation overhead* without affecting accuracy?
- How to flexibly *reduce the memory pressure* on mobile?

3.2 Core-Periphery Guided Self-Attention in ViT

The design of `ECP-ViT`, as shown in Figure 4, is inspired by Brain Neural Networks. Specifically, Figure 4 illustrates the selection of various CP graphs referenced in Figure 2 (a). The workflow involves CP graph generation, CP-guided self-attention, and CP-guided QKV multiplication, corresponding to Figure 2 (a), Figure 2 (b1), and Figure 2 (b2). In these networks, the core nodes maintain connections to all other core nodes, while edge nodes only connect to a subset (or empty) of the core nodes. We employ Grad-CAM to identify important regions of the images and assign the core nodes to those regions during training. Accordingly, the QKV matrices of these patches are divided into core and peripheral components. For example, for images with a resolution of 224x224 and a patch size of 16x16, there are a total of 196 patch tokens as nodes. For a core ratio of 10%, around 20 patch tokens are considered as cores, and we choose the top 20 important regions as cores. This partitioning method is inspired by human brain networks [60], where different networks exhibit different core ratios. This architecture allows BNNs to effectively enhance information transmission and communication for integrative processing [5, 19]. To incorporate the Core-Periphery principle into the self-attention mechanism of ViT, we redefined the self-attention operations based on the generated Core-Periphery (CP) graphs, where the patches are regarded as nodes, and the new self-attention relationships are represented by edges in the CP graph. Following this representation paradigm, a complete graph can depict the self-attention of the vanilla ViT. Similarly, the infusion of the Core-Periphery principle into the ViT architecture is achieved by enhancing the complete graph with the generated CP graphs effectively and conveniently. The new self-attention rules can then be redefined: CP graph can be represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes set $\mathcal{V} = \{\nu_1, ..., \nu_n\}$, edges set $\mathcal{E} \subseteq \{(\nu_i, \nu_j) | \nu_i, \nu_j \in \mathcal{V}\}$, and adjacency matrix A. The CP graph guided self-attention for a specific node i at r-th layer of ECP-ViT is defined as:

$$x_i^{(r+1)} = \sigma^{(r)}(\{(\frac{q_i^{(r)}(K_j^{(r)})^T}{\sqrt{d_k}})V_j^{(r)}, \forall j \in N(i)\}), \quad (1)$$

https://doi.org/10.52202/079017-3034

where $\sigma(\cdot)$ is the softmax function, $q_i^{(r)}$ is the query of patches in the i-th node in $\mathcal{G}$, $N(i) = \{i|i \vee (i,j) \in \mathcal{E}\}$ are the neighborhood nodes of node i, d_k is the dimension of queries and keys, and $K_j^{(r)}$ and $V_j^{(r)}$ are the key and value of patches in node j.

In ECP-ViT, each node can contain a single patch or a set of multiple patches. We propose the following patch-assigning pipeline to map the original patches to the nodes of the CP graph. In vanilla ViT with patch size 16×16, one input image with resolution 224×224 is divided into 196 patches. When we use a CP graph with n nodes to design the self-attention mechanism, 196 mod n nodes will be assigned $\lfloor 196/n \rfloor + 1$ patches and the remaining $n - (196 \ mod \ n)$ nodes will be assigned $\lfloor 196/n \rfloor$ patches. For example, if we use a 5 node CP graph, the 5 nodes will have 40, 39, 39, 39, and 39 patches, respectively; and if we use a 196 nodes CP graph in another case, each node will contain a single patch. Based on the above discussion, the CP graph-guided self-attention that is conducted at the node level can be formulated as:

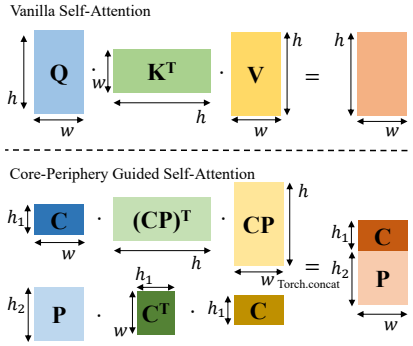

Figure 5: The Core-Periphery Principle guides Self-Attention, where the Query, Key, and Value matrices are partitioned into core (C) and periphery (P) components. The conventional self-attention mechanism is transformed into Core-Periphery (CP) attention through the guidance of Core-Periphery graphs.

$$\begin{aligned} &\text{CPAttention}(Q,K,V) = \\ &\quad \text{concat}(\text{softmax}(\frac{Q_c K^T}{\sqrt{d_k}})V, \text{softmax}(\frac{Q_p K_c^T}{\sqrt{d_k}})V_c), \end{aligned} \tag{2}$$

where queries, keys, and values of all patches are packed into matrices Q, K, and V, respectively, and subscript c and p represent the core parts and periphery parts of the matrices. The graphical illustration of CPAttention is shown in Figure 5. Similar to the multi-head attention in transformers [48], our proposed CP multi-head attention is formulated as:

$$\begin{aligned} &\text{MultiHead}(Q,K,V) = \text{concat}(head_1, ..., head_h)W^o, \\ &\text{where} \ \ head_i = \text{CPAttention}(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \tag{3}$$

where the parameter matrices W_i^Q, W_i^K, W_i^V and W^O are the projections. Multi-head attention helps the model to jointly aggregate information from different representation subspaces at various positions. In this work, we apply the CP principle to each representation subspace. Therefore, the self-attention guided by CP graphs in ECP-ViT reduces the computational budgets, while still maintaining the hardware-friendly computation pattern, i.e., converting the traditional self-attention into two small parts of matrix multiplication.

3.3 Flexible Data Layout Selection

DNN execution on mobile devices is in a manner of layer-wise computational graph (CG). The CG consists of nodes and edges, each node is an operator such as `MatMul` or `LayerNorm`, and the edge is an indicator to show the direction of the data flow. The main idea behind our optimization for eliminating data transformation is that, in a CG, each operator (such as `MatMul` or `Convolution`) has both producers and consumers. *The producer generates a layout based on the consumer's preferred data layout, resulting in relatively low additional overhead compared to explicitly reorganizing the data*. For instance, if a `MatMul` is followed by a `Transpose` and another `MatMul`, we can make the first `MatMul` directly generate the desired data layout for the second `MatMul`, thus avoiding the need for explicit data reorganization within `Transpose`. Our compiler optimizations consist of three steps: (i) Identify nodes to fuse; (ii) Determine possible data layouts for the key nodes; (iii) Evaluate possible data layouts. We elaborate on them in the following sections.

Identify nodes to eliminate starts by classifying the operator in CG into two types: 1) *key nodes* (nodes that perform the actual computation such as `MatMul`, `Softmax`, and `Add` are called *key*

nodes; and 2) nodes that only do layout transformation such as `Slice` and `Transpose` are called *transformative nodes*, which are the targets to eliminate. We first use the `transformative node` as a breaking point to partition the graph into a set of sub-graphs. We also apply the fusion rules similar to [37] to identify all fusion opportunities within the sub-graphs. To fully eliminate the data transformation operations, our general strategy is to find a common data layout which works efficiently for both contiguous sub-graphs. Note that, in this step, `transformative nodes` are still kept in the sub-graph because *key nodes'* data layouts are not the same.

Determine and evaluate possible data layouts are to find out the best intermediate data layout to eliminate the necessity of introducing additional operators solely for layout transformations. We conduct an exhaustive evaluation of all feasible data format selections, intending to optimize hardware accelerations. It is worth noting that varying data layouts result in distinct access patterns within the mobile GPU, as shown in Figure 6. Notably, noncontinuous data access patterns yield inferior data locality, thereby leading to increased latency. Our *dimension reduction heuristic* is aiming to find the reduction dimension(s) from both ends and group the reducing dimension continuously in the memory, in order to avoid the undesired data accessing pattern. For example, in a `Matmul` operation with $A_{m,k}$ and $B_{k,n}$, the k dimension is our reducing dimension.

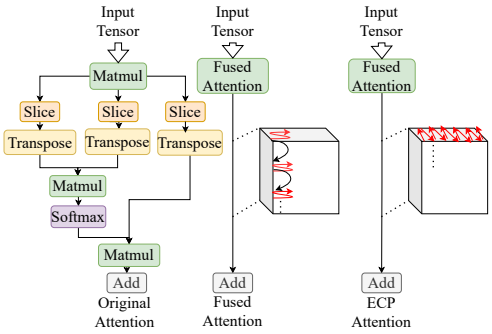

Figure 6: Data Access Pattern Comparisons. On the leftmost, it is the naive attention computational flow graph. In the middle part, it is the strided access pattern without any optimizations, and the rightmost graph is our optimized data access pattern.

Furthermore, specific operators in the subsequent operators may suitable for specific data formats. Take the `LayerNorm` operation as an example, it usually applies the calculation in one dimension only, we generally arrange the target dimension in the width continuously, and group other dimensions into the height dimension. It gives better data locality and GPU utilization. After using our heuristic algorithm to find the best data layout for each *key node*, we are then able to calculate the index transformation from the current node to its child nodes and map the index at the kernel level, which essentially eliminates all the trans-formative operations. Additionally, we store the whole intermediate results within the GPU texture memory to mitigate the time required for data transfer between the CPU and GPU during inter-operator operations. This adjustment is critical, given that data transfer speed between the GPU and CPU remains a persistent bottleneck in mobile device performance.

3.4 Compression-compilation co-design in ViT

We evaluate the speed of pruned models by utilizing compiler code generation and on-device latency measurement before performing a time-consuming pruning process. The compiler does not need absolute weight values for code generation and latency measurement, making the process more streamlined. Code generation with the compiler is much faster than Deep Neural Network (DNN) training. Therefore, before actually pruning the models, we use the compiler to evaluate them with different pruning ratios, using synthesized weights instead of real ones. This approach helps us establish a predictive curve that shows how pruning ratios correlate with expected latency. Such a strategy is crucial in determining the optimal pruning ratio that balances model performance and computational efficiency. It allows for a more effective decision on pruning before undertaking the computationally intensive DNN training process.

4 Evaluation

In this section, we evaluate the performance of our compiler-assisted framework with our *ECP-ViT* model deployed on mobile devices. We use CP-Level and core ratio interchangeably in the following section.

https://doi.org/10.52202/079017-3034

4.1 Setting

Models and datasets. The ECP-ViT is implemented based on the ViT architecture [17] and evaluated on 4 different datasets, STL-10 [12], CIFAR-100 [27], TinyImageNet [29], and ImageNet-1K [15]. TinyImageNet is a subset of ImageNet-1k containing 100,000 images distributed across 200 classes. The parameters of ECP-ViT were initialized and fine-tuned from ViT-B/16 trained on ImageNet-21K [28] for TinyImageNet and ImageNet, and used parameters from pre-trained ViT-S/16 for STL-10 and CIFAR-100. We trained the ECP-ViT for 50 epochs with batch size 256 for STL-10, CIFAR-100, and 128 for TinyImageNet and ImageNet-1K, and used AdamW optimizer and cosine learning rate schedule [35] with an initial learning rate of $5\mathrm{e}{-4}$ and minimum of $1\mathrm{e}{-7}$.

Evaluation environment. We compare the latency of ECP-ViT on off-the-shelf mobile devices against three state-of-the-art deep learning frameworks: Alibaba MNN [26], Tencent TNN, and Apache TVM [11].

We do not include emerging DNN inference frameworks like TFLite[2] and PyTorch Mobile[25] because they do not support ViT on mobile GPUs yet due to unsupported operators or insufficient resources (e.g., memory capacity). Our evaluation focuses on GPU instead of CPU or NPU for two reasons. Firstly, compared to mobile CPUs, mobile GPUs offer higher computational capacity with better power efficiency. Secondly, compared to mobile NPUs, the NPUs backend is often invoked via a system call provided by the Android Runtime System or specific hardware vendors which does not provide an interface for independent developers to support or optimize certain operators yet. We leave this as a future research direction.

The evaluations are conducted on a Oneplus 11 cell phone, which features a high-end Qualcomm Kryo octa-core CPU and a Qualcomm Adreno 740 GPU with 16 GB of unified memory. To demonstrate the portability of our methods, we also present results from testing on a low-end cell phone - Xiaomi 6 with limited memory and computation capacity, as shown in Figure 7. Xiaomi 6 is equipped with an ARM Octa-core CPU, an Adreno 540 GPU, and 6 GB of unified memory. We use 16-bit floating point precision across all frameworks and models on the mobile GPU. All latency data is collected from running the tests 50 times. However, since the variance is small, we only report averages.

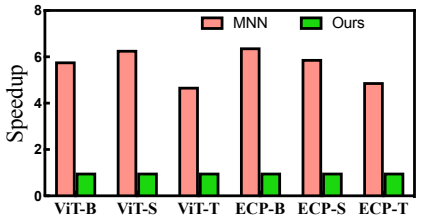

Figure 7: Latency Speedup Comparison Over Low-end Device (Xiao Mi 6). B, S, T short for Base, Small, and Tiny models, respectively.

4.2 Accuracy Comparison

The accuracy comparison with other works (with pruning and without pruning) is presented in Table 2. Notably, the accuracy of ECP-ViT is under the best ratio, showcasing its effectiveness. The core-periphery principle guided self-attention of ECP-ViT proves to be competitive with self-attention in a complete graph form. This demonstrates that the message exchange of core-periphery, derived from brain function networks, leverages the message communication in ViT.

We also assess the performance of the proposed ECP-ViT on ImageNet-1K with varying core ratios, and the results are detailed in Table 3. The baseline, denoted as vanilla ViT, featuring self-attention in a complete graph form, is considered to possess a core ratio of 1.0. Notably, for the ViT-base model scale, our ECP-ViT-base, integrating the core-periphery principle, demonstrates superior performance compared to ViT-base across a spectrum of core ratios ($0.7, 0.8$, and 0.9). ECP-ViT surpasses the baseline by 0.73% under a ratio of 0.9, suggesting that the sparsity of self-attention could enhance performance, with the core-periphery principle guiding self-attention proving to be an effective means of achieving this sparsity.

It's worth noting that our baseline is fine-tuned by us and already stands out compared to that reported in other works [65, 17, 47, 14]. Even for smaller model scales, such as ECP-ViT-small and ECP-ViT-tiny, the drop in accuracy is minimal when compared to vanilla ViT. Additionally, we conducted a comparative analysis of ECP-ViT with other competitive models on TinyImageNet, as depicted in Table 4. The results highlight the advantages of ECP-ViT in terms of prediction accuracy.

https://doi.org/10.52202/079017-3034

Table 2: Comparisons of results on the ImageNet dataset. All reported model results are based on pre-trained weights from ImageNet-21K. The accuracy, parameters, and MACs of ECP-ViT are under the best core ratio.

Model	Top-1 (%)	# Params.	# MACs
LTMP [8]	75.4	5.7M	1.5G
PVT-Tiny [50]	75.1	13.2M	1.97G
ECP-ViT-Tiny	75.8	5.83M	1.1G
PVT-Small [50]	79.8	24.5M	3.89G
ViT-S [17]	81.1	22M	4.62G
T2T-ViT$_t$-14 [63]	80.7	22M	4.8G
ECP-ViT-Small	80.9	21.7M	4.3G
ViT-Base/16 [17]	83.9	86.6M	17.6G
PVT-Large [50]	83.8	82.0M	11.84G
TNT-B [20]	84.1	66.0M	14.16G
DeiT-Base/16 [45]	84.2	86.6M	17.76G
ECP-ViT-Base	84.6	86.5M	16.96G

Table 3: Performance evaluation of ECP-ViT on ImageNet under different core ratios. We fine-tune the ECP-ViT using the pre-trained weights on ImageNet-21K. Top-1 accuracy is reported and shown in percentage. T, S, and B represents ECP-ViT-T, ECP-ViT-S, ECP-ViT-B, respectively.

Ratio	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0 (base)
T/16	65.50	69.14	69.39	69.75	71.38	73.13	74.55	75.48	75.84	75.90
S/16	73.24	74.42	74.98	76.32	77.82	78.78	79.82	80.34	80.88	81.12
B/16	76.15	77.98	79.40	80.51	81.80	83.22	84.03	84.51	84.62	83.89

Table 4: Comparison between ECP-ViT with other ViT variants on TinyImageNet. Top-1 Accuracy is shown in percentage. The best result of ECP-ViT under different core ratios is selected.

Model	Top1 Acc.	Params.	Image Res.
ViT-B/16 [17]	89.16	86.55M	224 $\times$ 224
DeiT-B/16 [46]	87.29	87.34M	224 $\times$ 224
BeiT-B/16 [4]	88.64	86.53M	224 $\times$ 224
ConViT-B/16 [14]	90.52	86.54M	224 $\times$ 224
ECP-ViT-B/16	**90.55**	86.50M	224 $\times$ 224

Table 5: Comparisons between ECP-ViT and vanilla ViT on STL10 and CIFAR100. The performance evaluation is under the best core ratio. Top-1 accuracy is reported and shown in percentage.

Model	STL-10	CIFAR-100
ViT-S/16 [17]	96.36	89.51
ECP-ViT-S/16	98.18 (+1.82)	89.56 (+0.05)

Furthermore, we extend our evaluation to STL10 and CIFAR-100, with detailed results provided in Table 5.

4.3 End-to-end Latency and Memory Comparison

Table 6 compares peak memory, latency, and cache miss rates between ViT-Base and ECP-ViT. Table 7 presents a comparison of latency among ECP-ViT, MNN, TNN, and TVM for vanilla ViT models. As the models retain their dense structure after CP pruning, they can still be executed on other frameworks. '-' means the model is not supported on the framework, due to lack of operator implementation or limited memory/computation resources. As shown in Table 7, compared to other state-of-the-art frameworks, ECP-ViT achieves an average speedup ranging from $4.8\times$ to $5.3\times$ for vanilla ViTs. This is because our compiler optimizations, such as data transformation elimination and operator fusion, help significantly reduce memory pressure and bandwidth demands. Moreover, our compiler optimizations for ECP-ViTs leverage the decreased computational complexity and yield a speedup ranging from $10.6\times$ to $16.1\times$ compared to vanilla ViT within our framework. Additionally, ECP-ViT outperforms other frameworks with a speedup ranging from $4.6\times$ to $26.9\times$. TVM exhibits

https://doi.org/10.52202/079017-3034

even higher latency for ECP-ViT due to the absence of systematic optimizations. The extra data transformation resulting from CP pruning leads to heightened demands on memory bandwidth. Our framework achieves real-time performance for small and tiny variants of ECP-ViT, meeting the requirement for real-time execution at 30 frames per second.

Table 6: Comparison of Peak Memory, Latency, and Miss Rates for ViT-Base and ECP-ViT.

Model	Peak Memory (MB)	Latency (ms)	Cache Miss Rate (%)		
			L1	L2	L3
ViT-Base[48]	454	421.25	0.77	5.94	15.12
ECP-ViT	403	99.84	0.66	5.38	15.05

Table 7: Latency comparison of 4 end-to-end frameworks on vanilla ViTs and CP-enabled ViTs using the GPU on Oneplus 11. We use CP level of 80% for all 3 variants (base, small and tiny). '-' means the models is not supported on the framework.

Model	TNN (ms)	TVM (ms)	MNN (ms)	**Ours (ms)**	**Speedup (avg)**
VIT-tiny	300.0	54.7	115.3	**17.6**	**4.8**
VIT-small	-	176.2	191.9	**37.7**	**4.9**
VIT-base	-	780.3	610.3	**131.9**	**5.3**
ECP-VIT-tiny	-	380.1	110.1	**15.2**	**16.1**
ECP-VIT-small	-	837.6	157.9	**31.1**	**16.0**
ECP-VIT-base	-	2033	563.2	**122.8**	**10.6**

5 Conclusion

This paper introduces ECP-ViT, a framework that enhances the deployment of Vision Transformer (ViT) models on mobile devices for real-time AI applications. Our approach includes a hardware-friendly pruning method inspired by the brain network and a set of compiler optimizations to eliminate data transformation bottlenecks. The results show that ECP-ViT not only reduces computational size but also improves prediction accuracy. It achieves up to 26.9x speedup compared to other state-of-the-art frameworks, enabling real-time performance on off-the-shelf mobile devices.

6 Impact Statement

This paper aims to advance the real-time implementation of brain-inspired AI on mobile devices. The integration of the brain-inspired core-periphery principle contributes to reducing the computation budget and enhancing prediction accuracy. Additionally, the optimization of hard-level data layout could significantly improve inference speed on mobile devices. This work has the potential to boost the development of brain-inspired AI on mobile devices.

Acknowledgment

We would like to express our gratitude to all those who contributed to this work, with special thanks to the constructive comments from the anonymous reviewers. This work was supported in part by the National Science Foundation (NSF) under the awards of CCF-2428108, OAC-2403090 . Any errors and opinions are not those of the NSF and are attributable solely to the author(s).

References

[1] Snapdragon 8 gen 2. https://www.topcpu.net/en/cpu/qualcomm-snapdragon-8-gen-2. Accessed: Jan-26-2024.

[2] Tensorflow lite. https://www.tensorflow.org/lite. Accessed: Jan-26-2024.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[5] Danielle S Bassett, Nicholas F Wymbs, M Puck Rombach, Mason A Porter, Peter J Mucha, and Scott T Grafton. Task-based core-periphery organization of human brain dynamics. *PLoS computational biology*, 9(9):e1003171, 2013.

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

[7] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *ArXiv*, abs/2306.12929, 2023.

[8] Maxim Bonnaerens and Joni Dambre. Learned thresholds token merging and pruning for vision transformers. *arXiv preprint arXiv:2307.10780*, 2023.

[9] Erik Cambria and Bebo White. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57, 2014.

[10] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021.

[11] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, 2018.

[12] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single layer networks in unsupervised feature learning. *AISTATS*, 2011.

[13] Arzu Çöltekin, Ian Lochhead, Marguerite Madden, Sidonie Christophe, Alexandre Devaux, Christopher Pettit, Oliver Lock, Shashwat Shukla, Lukáš Herman, Zdeněk Stachoň, et al. Extended reality in spatial sciences: A review of research challenges and future directions. *ISPRS International Journal of Geo-Information*, 9(7):439, 2020.

[14] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021.

[15] 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.

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

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

 https://doi.org/10.52202/079017-3034

[19] Shi Gu, Cedric Huchuan Xia, Rastko Ciric, Tyler M Moore, Ruben C Gur, Raquel E Gur, Theodore D Satterthwaite, and Danielle S Bassett. Unifying the notions of modularity and core–periphery structure in functional brain networks during youth. *Cerebral Cortex*, 30(3):1087–1102, 2020.

[20] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

[21] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 123–136. ACM, 2016.

[22] Zhe Huang, Xiaowei Yu, Dajiang Zhu, and Michael C. Hughes. Interlude: Interactions between labeled and unlabeled data to enhance semi-supervised learning. *International Conference on Machine Learning*, 2024.

[23] Rasheed Hussain and Sherali Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2018.

[24] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95, 2017.

[25] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021.

[26] Xiaotang Jiang, Huan Wang, Yiliu Chen, Ziqi Wu, Lichuan Wang, Bin Zou, Yafeng Yang, Zongyang Cui, Yu Cai, Tianhang Yu, Chengfei Lv, and Zhihua Wu. Mnn: A universal and efficient inference engine. In *MLSys*, 2020.

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[29] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[30] Royson Lee, Stylianos I Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D Lane. Mobisr: Efficient on-device super-resolution through heterogeneous mobile processors. In *The 25th annual international conference on mobile computing and networking*, pages 1–16, 2019.

[31] Zhengang Li, Geng Yuan, Wei Niu, Pu Zhao, Yanyu Li, Yuxuan Cai, Xuan Shen, Zheng Zhan, Zhenglun Kong, Qing Jin, et al. Npas: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14255–14266, 2021.

[32] Miaomiao Liu, Xuming He, and Mathieu Salzmann. Geometry-aware deep network for single-image novel view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4616–4624, 2018.

[33] Yichao Liu, Zongru Shao, and Nico Hoffmann. Global attention mechanism: Retain information to enhance channel-spatial interactions. *arXiv preprint arXiv:2112.05561*, 2021.

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

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

[36] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[37] Wei Niu, Jiexiong Guan, Yanzhi Wang, Gagan Agrawal, and Bin Ren. Dnnfusion: Accelerating deep neural networks execution with advanced operator fusion. PLDI 2021, page 883–898, New York, NY, USA, 2021. Association for Computing Machinery.

[38] Wei Niu, Xiaolong Ma, Sheng Lin, Shihao Wang, Xuehai Qian, Xue Lin, Yanzhi Wang, and Bin Ren. Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning. In *ASPLOS 2020*, pages 907–922, 2020.

[39] Jack Rae and Ali Razavi. Do transformers need deep long-range memory? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020. Association for Computational Linguistics.

[40] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[41] Aurko Roy*, Mohammad Taghi Saffar*, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers, 2020.

[42] Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. 2022.

[43] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12165–12174, 2022.

[44] 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.

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

[46] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357, 2021.

[47] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *In Computer Vision–ECCV 2022: 17th European Conference*, pages 459–479, 2022.

[48] 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.

[49] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European conference on computer vision*, pages 108–126. Springer, 2020.

[50] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.

[51] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[52] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

 https://doi.org/10.52202/079017-3034

[53] Zhenxiang Xiao, Yuzhong Chen, Junjie Yao, Lu Zhang, Zhengliang Liu, Zihao Wu, Xiaowei Yu, Yi Pan, Lin Zhao, Chong Ma, Xinyu Liu, Wei Liu, Xiang Li, Yixuan Yuan, Dinggang Shen, Dajiang Zhu, Dezhong Yao, Tianming Liu, and Xi Jiang. Instruction-vit: Multi-modal prompts for instruction learning in vision transforme. *Information Fusion*, (104):102204, 2024.

[54] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2964–2972, 2022.

[55] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 351–360, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[56] Hao Yu and Jianxin Wu. A unified pruning framework for vision transformers. *arXiv preprint arXiv:2111.15127*, 2021.

[57] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10819–10829, 2022.

[58] Xiaowei Yu, Zhang Lu, Haixing Dai, Lyu Yanjun, Lin Zhao, Tianming Liu, and Dajiang Zhu. Core-periphery principle guided redesign of self-attention in transformers. *arXiv preprint arXiv:2303.15569*, 2023.

[59] Xiaowei Yu, Zhang Lu, Tianming Liu, and Dajiang Zhu. Robust core-periphery constrained transformer for domain adaptation. *arXiv preprint arXiv:2308.13515*, 2023.

[60] Xiaowei Yu, Lu Zhang, Chao Cao, Tong Chen, Yanjun Lyu, Jing Zhang, Tianming Liu, and Dajiang Zhu. Gyri vs. sulci: Disentangling brain core-periphery functional networks via twin-transformer. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2024.

[61] Xiaowei Yu, Lu Zhang, Zihao Wu, and Dajiang Zhu. Core-periphery multi-modality feature alignment for zero-shot medical image analysis. *IEEE Transactions on Medical Imaging*, 2024.

[62] Xiaowei Yu, Lu Zhang, Zihao Wu, and Dajiang Zhu. Cp-clip: Core-periphery feature alignment clip for zero-shot medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2024.

[63] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567, 2021.

[64] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[65] Chuanyang Zheng, Kai Zhang, Zhi Yang, Wenming Tan, Jun Xiao, Ye Ren, and Shiliang Pu. Savit: Structure-aware vision transformer pruning via collaborative optimization. In *Advances in Neural Information Processing Systems*, pages 9010–9023, 2022.

[66] Mingjian Zhu, Yehui Tang, and Kai Han. Vision transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021.

[67] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

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: Main claims and brief results are in the abstract.

   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: [No]

   Justification: Due to space space limit, we only have a little discussion in the appendix.

   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: [No]

https://doi.org/10.52202/079017-3034

Justification: We do not have any theory to prove.

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 provided all the essential parts in 2.

   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: [No]

Justification: Because of copyright issues, we cannot make the framework public.

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: All training and test details are in 2 and 4

   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:

   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.

https://doi.org/10.52202/079017-3034

- 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 specified all the information 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: We have read and confirmed to conform with 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: [Yes]

    Justification: we put this section in 6.

    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: The paper has no such risks.

    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 have cited all the assets and works as required.

    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?

https://doi.org/10.52202/079017-3034

Answer: [NA]

Justification: No new assets.

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: No crowdsourcing

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.