ShiftAddViT: Mixture of Multiplication Primitives Towards Efficient Vision Transformer

Haoran You[*], **Huihong Shi**[*], **Yipin Guo**[*], and **Yingyan (Celine) Lin**
Georgia Institute of Technology, Atlanta, GA

Abstract

Vision Transformers (ViTs) have shown impressive performance and have become a unified backbone for multiple vision tasks. But both attention and multi-layer perceptions (MLPs) in ViTs are not efficient enough due to dense multiplications, resulting in costly training and inference. To this end, we propose to reparameterize the pre-trained ViT with a mixture of multiplication primitives, e.g., bitwise shifts and additions, towards a new type of multiplication-reduced model, dubbed **ShiftAddViT**, which aims for end-to-end inference speedups on GPUs without the need of training from scratch. Specifically, all `MatMuls` among queries, keys, and values are reparameterized by additive kernels, after mapping queries and keys to binary codes in Hamming space. The remaining MLPs or linear layers are then reparameterized by shift kernels. We utilize TVM to implement and optimize those customized kernels for practical hardware deployment on GPUs. We find that such a reparameterization on (quadratic or linear) attention maintains model accuracy, while inevitably leading to accuracy drops when being applied to MLPs. To marry the best of both worlds, we further propose a new mixture of experts (MoE) framework to reparameterize MLPs by taking multiplication or its primitives as experts, e.g., multiplication and shift, and designing a new latency-aware load-balancing loss. Such a loss helps to train a generic router for assigning a dynamic amount of input tokens to different experts according to their latency. In principle, the faster experts run, the larger amount of input tokens are assigned. Extensive experiments on various 2D/3D Transformer-based vision tasks consistently validate the effectiveness of our proposed ShiftAddViT, achieving up to **5.18**$\times$ latency reductions on GPUs and **42.9%** energy savings, while maintaining comparable accuracy as original or efficient ViTs.

1 Introduction

Vision Transformers (ViTs) have emerged as powerful vision backbone substitutes to convolutional neural networks (CNNs) due to their impressive performance [53, 16]. However, ViTs' state-of-the-art (SOTA) accuracy comes at the price of prohibitive hardware latency and energy consumption for both training on the cloud and inference on the edge, limiting their prevailing deployment and application on resource-constrained devices [32, 48], even though the pre-trained models are available. The bottlenecks are two folds based on many profiling results: attention and MLPs, e.g., they account for 36%/63% FLOPs and 45%/46% latency when running DeiT-Base [51] on NVIDIA Jetson TX2 GPU [67, 19]. Previous efficient ViT solutions mostly focus on macro-architecture designs [21, 37, 30, 51, 9, 57] and linear attention optimization [55, 47, 5, 13, 6, 68], while paying less attention on reducing the underlying dominate multiplications as well as co-optimizing both attention and MLPs.

We identify that one of the most effective yet still missing opportunities for accelerating ViTs is to reparameterize their redundant multiplications with a mixture of multiplication primitives, i.e.,

[*]Equal Contribution. Haoran proposed the idea, algorithm design, and drafted the paper; Huihong conducted algorithm experiments and measured accuracy; Yipin implemented TVM kernels and measured latency.

Preprint. Under review.

bitwise shifts and adds. The idea is drawn from the common hardware design practice in computer architecture or digital signal processing. That is, the multiplication can be replaced by bitwise shifts and adds [63, 22]. Such a hardware-inspired smart change leads to an efficient and fast hardware implementation without compromising the precision of results. As such, this paper embodies the shift&add idea in ViTs towards a new type of multiplication-reduced network. Moreover, unlike previous ShiftAddNet [65] which requires full and slow training and dedicated hardware accelerator supports, this work starts from pre-trained ViTs to avoid the tremendous cost of training from scratch and targets the end-to-end inference speedups, i.e., accelerating both attention and MLPs, on GPUs.

The seminal shift&add idea uniquely inspires us to design naturally hardware-friendly ViTs but leaves us with three challenges to be solved. *First*, how to effectively reparameterize ViTs with shifts and adds? Previous ShiftAddNet [65] reparameterizes CNNs by cascading shift layers and add layers, leading to a doubled number of layers or parameters. The customized CUDA kernels of shift and add layers also suffer from much slower training and inference than PyTorch [40] on GPUs. Both motivate a new easy-to-accelerate reparameterization method for ViTs. *Second*, how to maintain the accuracy after reparameterization? It is expected to see accuracy drops when it turns to shift and add primitives [8, 11]. Most works compensate for the accuracy drops by enlarging model sizes given the already extreme energy efficiency advantages of shifts and adds [65], e.g., up to 196× unit energy savings than multiplications as shown in Tab. 1. While for ViTs where images are split into non-overlapping tokens, one can uniquely leverage the nature of adaptive sensitivity among input tokens. In principle, the important tokens that contain targeting objects are expected to be processed with more powerful multiplications. Other tokens with unimportant background information will be left for cheaper primitives. Such a principle coincides with the spirit of recent token merging [3] and input adaptive methods [42, 64, 69] for ViTs. *Third*, how to balance the loading and processing time for sensitive and non-sensitive input tokens? For ViTs, the mixture of multiplication primitives results in dynamic processing speeds for sensitive and non-sensitive tokens that need to be balanced. Otherwise, the intermediate activations take a longer time to be synchronized before moving to the next layer. To the best of our knowledge, *this is the first attempt* to tackle the above three challenges and to embody the shift&add idea from the hardware field to design multiplication-reduced ViTs with a mixture of multiplication primitives. We achieve that goal by making the following contributions:

Table 1: Hardware cost under 45nm CMOS.

OPs	Format	Energy (pJ)	Area (μm^2)
Mult.	FP32	3.7	7700
	FP16	0.9	1640
	INT32	3.1	3495
	INT8	0.2	282
Add	FP32	1.1	4184
	FP16	0.4	1360
	INT32	0.1	137
	INT8	0.03	36
Shift	INT32	0.13	157
	INT16	0.057	73
	INT8	0.024	34

- We take inspiration from the hardware practice to reparameterize pre-trained ViTs with a mixture of complementary multiplication primitives, i.e., bitwise shifts and adds, to deliver a new type of multiplication-reduced network, dubbed **ShiftAddViT**. All `MatMuls` in attention are reparameterized by additive kernels, and the remaining linear layers and MLPs are reparameterized by shift kernels. The kernels are built in TVM for practical deployments.
- We develop a new mixture of experts (MoE) framework for ShiftAddViT to maintain accuracy after reparameterization. Each expert refers to multiplication or its primitives, e.g., multiplication and shift, that will be enabled depending on the importance of the given input token, e.g., multiplications for important tokens and shifts for unimportant ones.
- We propose to use a latency-aware load-balancing loss term in our MoE framework to assign a dynamic amount of input tokens to each expert. In this way, the assigned numbers of tokens are matched with experts' processing speeds, largely reducing the synchronization time.
- We conduct extensive experiments to validate the effectiveness of our proposed ShiftAddViT. Results on various 2D/3D Transformer-based vision models consistently show its superior efficiency, achieving up to **5.18**× latency reductions on GPUs and **42.9%** energy savings, while maintaining comparable or even higher accuracy than original ViTs.

2 Related Works

Vision Transformers (ViTs). ViTs [16, 51] split images as non-overlapping patches/tokens, outperforming CNNs on various vision tasks with a simple encoder-only architecture consisting of attention and MLPs. But the very first ViTs rely on costly pre-training on huge dataset [49]. DeiT [51] and T2T-ViT [70] avoid this by improving training recipes or designing new tokenization schemes. Later on, a surge of new ViT architecture designs emerges, e.g., PVT [56], CrossViT [7], PiT [25], MViT [18] and Swin-Transformer [33], for improving the accuracy-efficiency trade-offs with a pyramid-like architecture. Another trend of dynamic ViTs, e.g., DynamicViT [42], A-ViT [64], ToME [3], and MIA-Former [69], propose to adaptively identify and remove unimportant tokens. Our adopted MoE framework shares the same spirit but handles those tokens with cheaper multiplication primitives that can be run in parallel. Also, for deploying ViTs on the edge, LeViT [21], MobileViT [37], EfficientViT [5, 47] use efficient attention or incorporate CNN feature extractors, further enabling ViTs to run as fast as CNNs, e.g., MobileNets [44]. In contrast, our proposed ShiftAddViT *for the first time* takes inspiration from the shift&add hardware shortcuts to reparameterize the already pre-trained ViTs towards end-to-end inference speedups and energy savings without hurting the accuracy. Therefore, it is orthogonal to ViT backbone designs and can be applied on top of them.

Efficient ViTs. Tremendous efforts have been put into designing efficient ViTs, e.g., to tackle the costly self-attention module which has quadratic computational complexity with respect to the number of input tokens, linear attentions have been proposed and can be roughly categorized into two groups: local attention [33, 2, 52] or kernel-based linear attention [28, 13, 61, 36, 5, 32, 1]. For the former, Swin [33] performs similarity measurements among neighboring tokens instead of all tokens; QnA [2] shares the attention queries among all tokens; MaxViT [52] adopts block attention and takes dilated global attention into account for learning both local and global information. For the latter, most designs approximate the softmax function [13, 28, 4] or the full self-attention matrix [61, 36] with orthogonal features or kernel embeddings, then the computation order can be changed from $(\mathbf{QK})\mathbf{V}$ to $\mathbf{Q}(\mathbf{KV})$. Also, there are few works focusing on reducing the number of multiplications in ViTs. For example, Ecoformer [32] proposes a new binarization paradigm via kernelized hashing for $\mathbf{Q}/\mathbf{K}$ so that the `MatMuls` in attentions can be solely accumulations; Adder Attention [48] analyzes the difficulty of applying adder operations to attentions and proposes to include an additional identity mapping; ShiftViT [54] adopts spatial shift operations to attention modules. Different from all the above works, our ShiftAddViT focuses on not only attention but also MLPs towards end-to-end ViT speedups. All `MatMuls` can also be converted to additions/accumulations with simple quantization instead of relying on complex kernelized hashing (KSH) [32]. All remaining linear layers in attention or MLPs are converted to bitwise shifts or MoEs instead of spatial shifts as used in [54] for attentions.

Multiplication-less NNs. Many works target to reduce dense multiplications that dominate the time/energy cost of CNNs and ViTs. For CNNs, binary networks [14, 27] binarize both activations and weights, reducing multiplications to merely sign-flips; AdderNet [8, 62, 58] fully replaces multiplications with additions with only a slight accuracy drop; Shift-based networks use either spatial shifts [60] or bitwise shifts [17] to replace multiplications. Recently, some ideas originated from CNNs have also been applied to ViTs. For example, BiTs [34, 24] enables binary transformers at a decent accuracy; Shu et al. and Wang et al. also migrate the add or shift idea to ViTs with the scope limited to attentions [48, 54]. Apart from network designs, there are also neural architecture search efforts made towards both accurate and efficient networks [66, 29]. As compared to the most related work in literature, i.e., ShiftAddNet [65] and ShiftAddNAS [66], our ShiftAddViT *for the first time* enables shift&add idea in ViTs without compromising the model accuracy, featuring three more characteristics that make it more suitable for practical usage and deployment: (1) starting from pre-trained ViTs to avoid the costly and tedious training from scratch; (2) focusing on optimization on GPUs apart from dedicated hardware acceleration on FPGAs/ASICs; (3) seamlessly compatible with the MoE framework to enable switching between the mixture of multiplication primitives, such token-level routing and parallelism are uniquely applicable and designed for ViTs.

3 Preliminaries

Self-attention and Vision Transformers. Self-attention is a core ingredient of the Transformer [53, 16], and usually contains multiple heads H with each measuring pairwise correlations among all input tokens to capture global-context information. It can be defined as below:

$$\texttt{Attn}(\mathbf{X}) = \texttt{Concat}(\mathrm{H}_1, \cdots, \mathrm{H}_h)\mathbf{W}_O, \text{ where } \mathrm{H}_i = \texttt{Softmax}\left(\frac{f_Q(\mathbf{X}) \cdot f_K(\mathbf{X})^T}{\sqrt{d_k}}\right) \cdot f_V(\mathbf{X}), \quad (1)$$

where h denotes the number of heads. Within each head, input tokens $\mathbf{X} \in \mathbb{R}^{n \times d}$ of length n and dimension d will be linearly projected to query, key, and value matrices, i.e., $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d_k}$, through three linear mapping functions, $f_Q = \mathbf{XW}_Q, f_K = \mathbf{XW}_K, f_V = \mathbf{XW}_V$, where $d_k = d/h$ is the embedding dimension of each head. The results from all heads are concatenated and projected

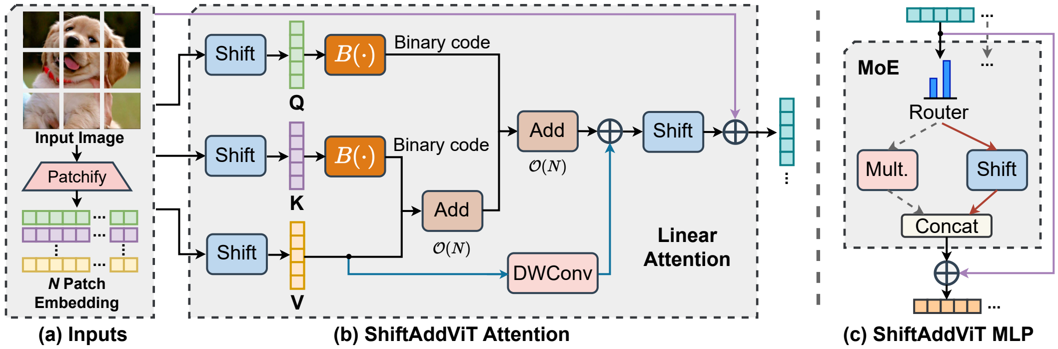


Figure 1: Illustration of the network architecture overview of ShiftAddViT.

with a weight matrix $\mathbf{W}_O \in \mathbb{R}^{d\times d}$ to generate the final outputs. Such an attention module is followed by MLPs with residuals to construct one transformer block that can be formulated as below:

$$\mathbf{X}_{\texttt{Attn}} = \texttt{Attn}(\texttt{LayerNorm}(\mathbf{X})) + \mathbf{X}, \quad \mathbf{X}_{\texttt{MLP}} = \texttt{MLP}(\texttt{LayerNorm}(\mathbf{X}_{\texttt{Attn}})) + \mathbf{X}_{\texttt{Attn}}. \tag{2}$$

Shift and Add Primitives. Multiplications are inefficient if being directly implemented in hardware. Shift and add hardware primitives server as "shortcuts" for efficient hardware designs. For shift operations, they are equivalent to multiplying by power-of-two. As summarized in Tab. 1, such bitwise shifts can save up to 23.8× unit energy and 22.3× chip area as compared to multiplications [31, 65] when considering the `INT32` data format and a 45nm CMOS technology. For add operations, they are another kind of efficient primitives and achieve as high as 31.0× energy cost and 25.5× area savings as compared to multiplications. Both of them enable many network architecture designs [8, 17, 65, 66].

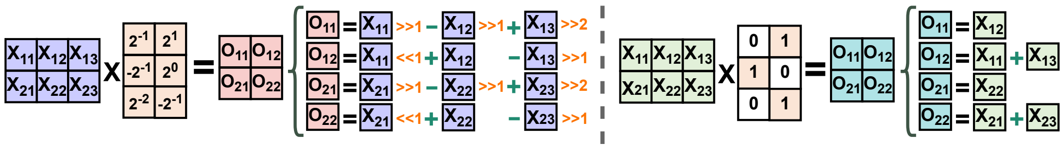
4 The Proposed ShiftAddViT

Overview. Given the widely available pre-trained ViT model weights, our goal is to finetune and convert them to multiplication-reduced ShiftAddViTs for reducing runtime latency and improving energy efficiency. To achieve that, we need to reparameterize both attentions and MLPs in ViTs to shift and add operations. Instead of adopting cascaded shift layers and add layers like ShiftAddNet [65] or designing specific forward and backpropagation schemes as AdderNet [8, 48], we adhere to the simple but effective spirit of ViTs' long-dependency modeling capability and propose to replace the dominate MLPs/Linears/`MatMuls` with shifts and adds while keeping the attention scheme untouched. It allows ShiftAddViTs to be adapted from existing ViTs without training from scratch. As shown in Fig. 1, for attentions, we convert four Linear layers and two `MatMuls` to shift and add layers, respectively; For MLPs, directly replacing with shift layers leads to severe accuracy drops, we consider designing a tailored MoE framework with a mixture of multiplication primitives, e.g., multiplication and shift, towards both accurate and efficient final models. The above two steps transfer pre-trained ViTs to ShiftAddViTs with much reduced runtime latency while comparable or even higher accuracy.

4.1 ShiftAddViT: Reparameterization Towards Multiplication-reduced Networks

This subsection describes how we reparameterize pre-trained ViTs with hardware-efficient shift and add operations, including a detailed implementation and the corresponding sensitivity analysis.

Reparameterization of Attention. There are two categories of layers in attention modules: `MatMuls` and Linears, that can be converted to `Add` and `Shift` layers, respectively. In order to reparameterize `MatMuls` to `Add` layers, we consider performing binary quantization on one operand during `MatMuls`, e.g., $\mathbf{Q}$ or $\mathbf{K}$ for the `MatMul` of $\mathbf{QK}$, so that the multiply-and-accumulate (MAC) operations between two matrices will be replaced with merely energy-efficient add operations [32]. Furthermore, to build ShiftAddViT on top of efficient linear attentions [55, 32, 36], we exchange the order of `MatMuls` from $(\mathbf{QK})\mathbf{V}$ to $\mathbf{Q}(\mathbf{KV})$ for achieving linear complexity w.r.t. the number of input tokens. In this way, the binary quantization will be applied to $\mathbf{K}$ and $\mathbf{Q}$ as illustrated in Fig. 1 (b), leaving the more sensitive $\mathbf{V}$ branch as high precision. This also allows us to insert a lightweight depthwise convolution (DWConv) to $\mathbf{V}$ branch in parallel to enhance local feature capturing capability with negligible overhead (<1% of total MACs) following recent SOTA linear attention designs [61, 68]. On the other hand, for reparameterizing the left four Linears in the attention module with `Shift` layers, we resort to sign flips and power-of-two shift parameters to represent the Linear weights [17, 65].



(a) Shift Layers (b) Add Layers

Figure 2: Illustration of both `Shift` and `Add` layers, where $\mathbf{X}$ and $\mathbf{O}$ refer to inputs and outputs.

We illustrate the implementation of `Shift` and `Add` layers in Fig. 2 and formulate them as follows:

$$\mathbf{O}_{\texttt{Shift}} = \sum \mathbf{X}^{\mathbf{T}} \cdot \mathbf{W_S} = \sum \mathbf{X}^{\mathbf{T}} \cdot \mathbf{s} \cdot 2^{\mathbf{P}}, \quad \mathbf{O}_{\texttt{Add}} = \sum \mathbf{X}^{\mathbf{T}} \cdot \mathbb{1}\{\mathbf{W_A} \neq \mathbf{0}\}, \tag{3}$$

where $\mathbf{X}$ refers to input activations, $\mathbf{W_S}$ and $\mathbf{W_A}$ are weights in `Shift` and `Add` layers, respectively. For `Shift` weights $\mathbf{W_S} = \mathbf{s} \cdot 2^{\mathbf{P}}$, $\mathbf{s} \in \{-1, 1\}$ denotes sign flips, $2^{\mathbf{P}}$ represents bitwise shift towards left ($\mathbf{P} > 0$) or right ($\mathbf{P} < 0$). For `Add` weights, we have applied binary quantization and will skip zeros directly, the remaining weights correspond to additions or accumulations. Combining those two weak players, i.e., `Shift` and `Add`, leads to higher representation capability as also stated in ShiftAddNet [65], we highlight that our ShiftAddViT seamlessly achieves such combination for ViTs, without the necessity of cascading both of them as a substitute for one convolutional layer.

Reparameterization of MLPs. MLPs are often overlooked but also dominate ViTs' runtime latency apart from attentions, especially for the small or medium amount of tokens [19, 73, 41]. One natural thought is to reparameterize all MLPs with `Shift` layers as defined above. However, such aggressive design indispensably leads to severe accuracy drops, e.g., $\downarrow$1.18% for PVTv1-T [56], (DWConv used between two MLPs in PVTv2 are kept). We resolve this by proposing a new MoE framework to achieve speedups with comparable accuracy, as shown in Fig. 1 (c) and will be elaborated in Sec. 4.2.

Sensitivity Analysis. To investigate whether the above reparameterization methods work as expected, we break down all components and conduct sensitivity analysis. Each of them is applied to ViTs with only 30 epochs of finetuning. As summarized in Tab. 2, applying linear attention (`LA`), `Add`, or `Shift` to attention layers makes little influence on accuracy. But applying `Shift` to MLPs may lead to severe accuracy drops for sensitive models like PVTv1-T. Also, making MLPs more efficient contribute a lot to the energy efficiency (e.g., occupy 65.7% on PVTv2-B0). Both the accuracy and efficiency perspectives motivate us to propose a new reparameterization scheme for MLPs.

Table 2: Sensitivity analysis of reparameterizing ViTs with `Shift` and `Add`.

Components	Apply	Acc. (%) of PVT [56, 57]	
		PVTv2-B0	PVTv1-T
-	-	70.50	75.10
-	MSA	71.25	76.21
Attention	LA+Add	70.95	75.20
	Shift	70.96	76.05
MLPs	Shift	70.28	73.92
	MoE	70.86	74.81

4.2 ShiftAddViT: Mixture of Experts Framework

This subsection elaborates on the hypothesis and method of our ShiftAddViT MoE framework.

Hypothesis. We hypothesize there are important yet sensitive input tokens that necessitate the more powerful networks otherwise suffer from accuracy drops. Those tokens are likely to contain object information that directly correlates with our task objectives as validated by [3, 69, 42, 64]. While the left tokens are less sensitive and can be fully represented even using cheaper `Shift` layers. Such an input-adaptive nature of ViTs calls for a framework featuring a mixture of multiplication primitives.

Mixture of Multiplication Primitives. Motivated by the sensitivity analysis in Sec. 4.1, we consider two experts `Mult.` and `Shift` for processing important yet sensitive tokens and insensitive tokens, respectively. As illustrated in Fig. 1 (c), we apply the MoE framework to compensate for the accuracy drop of `Shift`. Each input token representation x will be passed to one expert according to the gate values p_i in routers, which are normalized via a softmax distribution $p_i(\mathbf{x}) := e^{p_i} / \sum_j^n e^{p_j}$ and will be jointly trained with model weights. The output can be formulated as $\mathbf{y} = \sum_i^n G(\mathbf{x}) \cdot E_i(\mathbf{x})$, where gate function $G(\mathbf{x}) = p_i(\mathbf{x}) \cdot \mathbb{1}\{p_i(\mathbf{x}) \geq p_j(\mathbf{x}), \forall j \neq i\}$, n and E_i denote the number of experts and i-th expert, respectively. Specifically, there are two obstacles when implementing such MoE in TVM: (1) dynamic input allocation, and (2) parallel computing. For (1), we follow Nimble [46] to handle dynamic shapes; For (2), we perform modularized optimization in pursuit of parallelism.

Latency-aware Load-balancing Loss. The key to our MoE framework is the design of a routing function to balance all experts towards higher accuracy and lower latency. Previous solutions [20, 23] use homogeneous experts and treat them equally. While in our MoE framework, the divergence and heterogeneity between powerful yet slow `Mult.` and fast yet less powerful `Shift` experts incur one unique question: *How to orchestrate the workload of each expert to reduce the synchronization time?* Our answer is a latency-aware load-balancing loss, which ensures (1) all experts receive the expected weighted sums of gate values; and (2) all experts are assigned the expected number of input tokens. Those two conditions are enforced by adopting importance and load balancing loss as defined below:

$$\mathcal{L}_{\texttt{IMP}} = \texttt{SCV}\left(\{\alpha_i \cdot \sum\nolimits_{\mathbf{x}\in\mathbf{X}} p_i(\mathbf{x})\}_{i=1}^{n}\right), \quad \mathcal{L}_{\texttt{LOAD}} = \texttt{SCV}\left(\{\alpha_i \cdot \sum\nolimits_{\mathbf{x}\in\mathbf{X}} q_i(\mathbf{x})\}_{i=1}^{n}\right), \tag{4}$$

where SCV denotes the squared coefficient of variation of given distributions over experts (also used in [43, 45]); α_i refers to the latency-aware coefficient of i-th expert and is defined by $^{(\texttt{Lat}_i)}/_{(\sum_j \texttt{Lat}_j)}$ because that the expected assignments are inversely proportional to runtime latency of i-th expert, $\texttt{Lat}_i$; $q_i(\mathbf{x})$ is the probability that the gate value of i-th expert outweighs the others, i.e., top1 expert, and is given by $\mathbf{P}\left(p_i(\mathbf{x}) + \epsilon \geq p_j(\mathbf{x}), \forall j \neq i\right)$. Note that this probability is discrete, we use a noise proxy ϵ following [45, 23] to make it differentiable. The above latency-aware importance loss and load-balanced loss help to balance gate values and workload assignments, respectively, and can be integrated with classification loss $\mathcal{L}_{\texttt{CLS}}$ as the total loss function $\mathcal{L}(\mathbf{X}) = \mathcal{L}_{\texttt{CLS}}(\mathbf{X}) + \lambda \cdot (\mathcal{L}_{\texttt{IMP}}(\mathbf{X}) + \mathcal{L}_{\texttt{LOAD}}(\mathbf{X}))$ for training ViTs and gates simultaneously. λ is set as 0.01 for all experiments.

5 Experiments

5.1 Experiment Settings

Models and Datasets. *Tasks and Datasets.* We consider two representative 2D and 3D Transformer-based vision tasks to demonstrate the superiority of the proposed ShiftAddViT, including 2D image classification on ImageNet dataset [15] with 1.2 million training and 50K validation images and 3D novel view synthesis (NVS) task on Local Light Field Fusion (LLFF) dataset [38] with eight scenes. *Models.* For the classification task, we consider PVTv1 [56], PVTv2 [57], and DeiT [51]. For the NVS task, we consider Transformer-based GNT [50] with View- and Ray-Transformer models.

Training and Inference Details. *For the classification task,* we follow Ecoformer [32] to initialize the pre-trained ViTs with Multihead Self-Attention (MSA) weights, based on which we apply our reparameterization a two-stage finetuning: (1) convert MSA to linear attention [68] and reparameterize all `MatMuls` with add layers with 100 epoch finetuning, and (2) reparameterize MLPs or linear layers with shift or MoE layers after finetuning another 100 epoch. Note that we follow PVTv2 [57] and Ecoformer to keep the last stage as MSA for fair comparisons. *For the NVS task,* we still follow the two-stage finetuning but do not convert MSA weights to linear attention to maintain the accuracy. All experiments are run on a server with eight RTX A5000 GPUs with each having 24GB GPU memory.

Baselines and Evaluation Metrics. *Baselines.* For the classification task, we reparameterize and compare our ShiftAddViT with PVTv1 [56], PVTv2 [57], Ecoformer [32], and their MSA variants. For the NVS task, we reparameterize on top of GNT [50] and compare with both vanilla NeRF [39] and GNT [50]. *Evaluation Metrics.* For the classification task, we evaluate the ShiftAddViT and baselines in terms of accuracy, GPU latency and throughput measured on an RTX 3090 GPU. For the NVS task, we evaluate all models in terms of PSNR, SSIM [59], and LPIPS [71]. We also measure and report the energy consumption of all the above models based on an Eyeriss-like hardware accelerator [12, 72], which calculates not only computational but also data movement energy.

5.2 ShiftAddViT over SOTA Baselines on 2D Tasks

Table 3: Overall comparison between ShiftAddViT and the most competitive baseline on five models.

Models	Methods	Acc. (%)	Latency (ms)	Energy (mJ)
PVTv2-B0	Ecoformer [32]	70.44	7.82	33.64
	ShiftAddViT	**70.59**	**1.51**	**27.13**
PVTv1-T	Ecoformer [32]	NaN	7.43	93.47
	ShiftAddViT	**74.93**	**1.97**	**72.59**
PVTv2-B1	Ecoformer [32]	78.38	8.02	106.2
	ShiftAddViT	**78.49**	**2.49**	**85.34**
PVTv2-B2	Ecoformer [32]	81.28	15.43	198.2
	ShiftAddViT	**81.32**	**4.83**	**163.9**
DeiT-T	MSA [51]	72.20	5.12	66.88
	ShiftAddViT	**72.40**	**2.94**	**38.21**

To evaluate the effectiveness of our proposed techniques, we apply the ShiftAddViT idea to five commonly used ViT models, including DeiT [51] and various variants of PVTv1 [56] and PVTv2 [57]. We compare their performance with baselines on the image classification task. Tab. 3 highlights the overall comparison with the most competitive baseline, Ecoformer [32], from which we see that ShiftAddViT consistently outperforms all baselines in terms of accuracy-efficiency tradeoffs, achieving **1.74**$\times \sim$ **5.18**$\times$ latency reduction on GPUs and **19.4**% $\sim$ **42.9**% energy savings measured on the Eyeriss accelerator [12] with comparable or even better accuracy ($\uparrow$**0.04**% $\sim$ $\uparrow$**0.20**%). Note

Table 4: Comparisons between ShiftAddViT and baselines. We show the breakdown analysis of ShiftAddViT with two kinds of $\mathbf{Q}/\mathbf{K}$ quantization. The throughput or latency is measured with a batch size of 32 or 1, where † denotes that the numbers are measured and reported after optimizing ShiftAddViT using TVM. We report both real-device and modularized latency for models with MoE.

Methods	Linear Attn	Add KSH	Add Quant.	Shift	MoE	PVTv2-B0 [57] Acc. (%)	PVTv2-B0 [57] Lat. (ms)	PVTv2-B0 [57] T. (img./s)	PVTv1-T [56] Acc. (%)	PVTv1-T [56] Lat. (ms)	PVTv1-T [56] T. (img./s)
MSA	✗	✗	✗	✗	✗	70.77	4.62	989	76.21	4.73	903
PVT [57]	✓	✗	✗	✗	✗	70.50	6.25	2227	75.10	5.78	1839
PVT+MoE	✓	✗	✗	✗	✓ (MLPs)	70.82	12.46	1171	75.27	10.91	834
Ecoformer [32]	✓	✓	✗	✗	✗	70.44	7.82	1348	NaN	7.43	1021
ShiftAddViT (with KSH [32] or Quant. [26] to binarize Q/K)	✓	✗	✗	✗	✗	71.19	6.13	2066	75.50	5.78	1640
	✓	✓	✗	✗	✗	70.95	1.07†	2530†	75.20	1.42†	1683†
	✓	✓	✗	✓ (Attn)	✗	70.53	1.04†	2447†	74.77	1.39†	1647†
	✓	✓	✗	✓ (Attn)	✓ (MLPs)	70.16	1.39†/1.11*	N/A	74.44	1.91†/1.21*	N/A
	✓	✓	✗	✗	✓ (Both)	70.38	1.59†/1.20*	N/A	74.73	2.12†/1.21*	N/A
	✓	✗	✗	✗	✗	71.36	6.34	2014	75.64	5.48	1714
	✓	✗	✓	✗	✗	71.04	1.00†	2613†	75.18	1.20†	1907†
	✓	✗	✓	✓ (Both)	✗	68.57	0.97†	2736†	73.47	1.18†	1820†
	✓	✗	✓	✗	✓ (Both)	70.59	1.51†/1.12*	N/A	74.93	1.97†/1.02*	N/A

* denotes the modularized latency simulated by separately optimizing each expert/router with ideal parallelism.

Table 5: Comparisons between ShiftAddViT and baselines. We apply our Shift&Add techniques on top of SOTA Transformer-based NeRF model, GNT [50], and report averaged PSNR (↑), SSIM (↑) and LPIPS (↓) on the LLFF dataset [38] across eight scenes. In addition, we also show the results on two representative scenes, Orchids, and Flower, as well as the rendering latency and energy measured on an Eyeriss-like accelerator. More results for other scenes are available in Appendix.

Methods	Add	Shift	MoE	LLFF Averaged PSNR	LLFF Averaged SSIM	LLFF Averaged LPIPS	Orchids PSNR	Orchids SSIM	Orchids LPIPS	Flower PSNR	Flower SSIM	Flower LPIPS	Lat. (s)	Energy (J)
NeRF [39]	-	-	-	26.50	0.811	0.250	20.36	0.641	0.321	27.40	0.827	0.219	683.6	1065
GNT [50]	-	-	-	27.24	0.889	0.093	20.67	0.752	0.153	27.32	0.893	0.092	1071	1849
ShiftAddViT	✓	✗	✗	26.85	0.874	0.116	20.74	0.730	0.182	28.02	0.891	0.089	1108	1697
	✓	✓(Both)	✗	26.85	0.875	0.116	20.78	0.730	0.182	28.05	0.892	0.088	568.5	844.0
	✓	✓(Attn)	✓(MLPs)	26.92	0.876	0.114	20.73	0.731	0.180	28.20	0.894	0.087	746.6	1093
	✗	✓(Both)	✗	27.05	0.881	0.107	20.84	0.746	0.169	28.14	0.896	0.083	531.2	995.6

that we apply MoE to both linear and MLP layers. Moreover, we further provide comparisons of a variety of ShiftAddViT variants and PVT with or without multi-head self-attention (MSA) and MoE in Tab. 4 and 6. We observe that our ShiftAddViT can be seamlessly integrated with linear attention [68, 61], simple or advanced $\mathbf{Q}/\mathbf{K}$ quantization [32, 26], and the proposed MoE framework, achieving up to **3.06**×/**4.14**×/**8.25**× and **2.47**×/**1.10**×/**2.09**× latency reductions and throughput improvements after the TVM optimization on an RTX 3090 as compared to MSA, PVT, and PVT+ MoE (note: two `Mult.` experts) baselines, respectively, under comparable accuracies, i.e., ±0.5%.

5.3 ShiftAddViT over SOTA Baselines on 3D Tasks

We also extend our methods to 3D NVS tasks and built ShiftAddViT on top of Transformer-based GNT models [50]. We test and compare the PSNR, SSIM, LPIPS, latency, and energy of both ShiftAddViT and baselines, i.e., vanilla NeRF [39] and GNT [50], on the LLFF dataset across eight scenes. As shown in Tab. 5, our ShiftAddViT consistently leads to better accuracy-efficiency tradeoffs, achieving **22.3%/50.4%** latency reductions and **20.8%/54.3%** energy savings under comparable or even better generation quality (↑0.55/↓0.19 averaged PSNR across eight scenes), as compared to NeRF and GNT baselines, respectively. Note that GNT costs more than NeRF because of an increased number of layers. In particular, ShiftAddViT even achieves better generation quality (up to ↑0.88 PSNR) than GNT on some representation scenes, e.g., Orchid and Flower, as highlighted in Tab. 5, where the first, second, and third ranking performance are noted with corresponding colors. The full results of eight scenes are supplied in Appendix. Note that we only finetune ShiftAddViT for 140K steps on top of pre-trained GNT models instead of training up to 840K steps from scratch.

5.4 Ablation Studies of ShiftAddViT

Performance Breakdown Analysis. To investigate how each of the proposed techniques contributes to the final performance, we conduct ablation studies among various kinds of ShiftAddViT variants on PVTv1-T, PVTv2-B0/B1/B2 models to gradually break down and show the advantage of each component. As shown in Tab. 4 and 6, we have three general observations: (1) ShiftAddViT is robust to the binarization method of $\mathbf{Q}/\mathbf{K}$, e.g., it is compatible with either KSH [32] or vanilla binarization [26], achieving on average **3.03**× latency reductions than original PVT under comparable

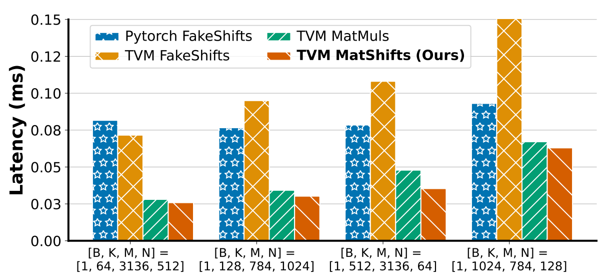


Figure 3: MLP/Linear latency speedups using shifts, where inputs are of shape $B \times K \times M$, weights are of shape $K \times N$. All dimensions are set w.r.t. the real dimensions in PVTs [56].

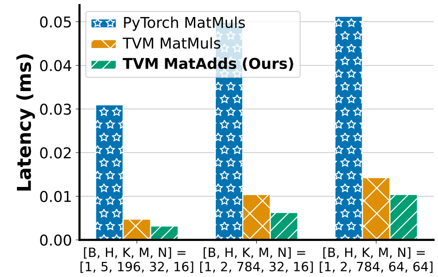


Figure 4: `MatMuls` latency speedups using adds, where inputs are of shape $B \times H \times K \times M$, weights are of shape $B \times H \times K \times N$.

Table 6: Comparisons between ShiftAddViT and baselines on PVTv2-B1 and PVTv2-B2. The throughput or latency is measured with a batch size of 32 or 1 on an RTX 3090 GPU.

Methods	Linear Attn	Add KSH	Add Quant.	Shift	MoE	PVTv2-B1 [57] Acc. (%)	PVTv2-B1 [57] Lat. (ms)	PVTv2-B1 [57] T. (img./s)	PVTv2-B2 [57] Acc. (%)	PVTv2-B2 [57] Lat. (ms)	PVTv2-B2 [57] T. (img./s)
MSA	✗	✗	✗	✗	✗	78.83	4.77	821	81.82	9.08	508
PVT [57]	✓	✗	✗	✗	✗	78.70	6.75	1512	82.00	12.20	921
Ecoformer [32]	✓	✓	✗	✗	✗	78.38	8.02	874	81.28	15.43	483
ShiftAddViT (with KSH [32] or Quant. [26] to binarize Q/K)	✓	✗	✗	✗	✗	78.80	6.36	1373	81.40	9.21	796
	✓	✓	✗	✗	✗	78.70	$1.70^{\dagger}$	$923^{\dagger}$	81.31	$3.18^{\dagger}$	$518^{\dagger}$
	✓	✓	✗	✓ (`Attn`)	✓ (`MLPs`)	78.20	$2.46^{\dagger}/1.46^{*}$	N/A	81.06	$4.17^{\dagger}/3.22^{*}$	N/A
	✓	✓	✗	✗	✓ (`Both`)	78.38	$2.64^{\dagger}/1.48^{*}$	N/A	81.32	$4.83^{\dagger}/3.35^{*}$	N/A
	✓	✗	✗	✗	✗	78.93	6.49	1327	81.55	11.82	782
	✓	✗	✓	✗	✗	78.70	$1.57^{\dagger}$	$942^{\dagger}$	81.33	$2.82^{\dagger}$	$553^{\dagger}$
	✓	✗	✓	✓ (`Both`)	✗	77.55	$1.54^{\dagger}$	$1021^{\dagger}$	79.97	$2.93^{\dagger}$	$600^{\dagger}$
	✓	✗	✓	✗	✓ (`Both`)	78.49	$2.49^{\dagger}/1.35^{*}$	N/A	81.01	$4.55^{\dagger}/3.16^{*}$	N/A

accuracies ($\pm 0.5\%$), and both **3.29**$\times$/**1.32**$\times$ latency/energy reductions and 0.04% $\sim$ 0.20% accuracy improvements over Ecoformer [32]; (2) vanilla binarization methods work better than KSH in our ShiftAddViT framework in terms of both accuracy ($\downarrow$0.05% $\sim$ $\uparrow$0.21%) and efficiency (on average $\downarrow$5.9% latency reductions and $\uparrow$5.8% throughput boosts). Moreover, KSH requires that $\mathbf{Q}$ and $\mathbf{K}$ are identical while vanilla binarization [26] does not have such a limitation and leads to on average $\uparrow$0.15% accuracy of adopting linear attention. Also note that our adopting linear attention following [68] works better ($\uparrow$0.29%) than PVT [56, 57]; (3) Replacing linear layers in attention modules or `MLPs` with `Shift` layers lead to an accuracy drop while our proposed MoE can help to compensate for it. Specifically, adopting `Shift` layers leads to on average 1.67% accuracy drop while MoE instead improves on average 1.37% accuracy. However, MoE hurts the efficiency of both the PVT baseline and our ShiftAddViT without customized system acceleration, e.g., PVT+MoE increases 94% latency and reduces 51% throughput than PVT on GPUs, due to limited parallelism supports, especially for TVM. We report the modularized latency by separately optimizing each expert to demonstrate the potential of MoE's double-winning accuracy ($\uparrow$0.94% $\sim$ $\uparrow$2.02%) and efficiency ($\downarrow$15.4% $\sim$ $\uparrow$42.7%).

Apart from 2D tasks, we also report the performance breakdown on 3D NVS tasks, as shown in Tab. 5. The above three observations still hold except for the `Shift` layer, as we find that ShiftAddViT with all linear layers or `MLPs` replaced by `Shift` layers achieve even better PSNR ($\uparrow$0.13 $\sim$ $\uparrow$0.20), SSIM ($\uparrow$0.005 $\sim$ $\uparrow$0.007), and LPIPS ($\downarrow$0.007 $\sim$ $\downarrow$0.009) than other variants. In addition, we profile on an Eyeriss accelerator to show the energy breakdown of both ShiftAddViT and baselines. As shown in Fig. 5, our ShiftAddViT reduced **42.9**% and **40.9**% energy on top of DeiT-T and GNT, respectively. Among them applying `Add` layers lead to **93.8**% and **63.8**% energy reductions than original `MatMuls`, `Shift` layers help to reduce **28.5**% and **37.5**% energy than previous linear/MLP layers. This set of experiments validates the effectiveness of each component in our proposed ShiftAddViT framework.

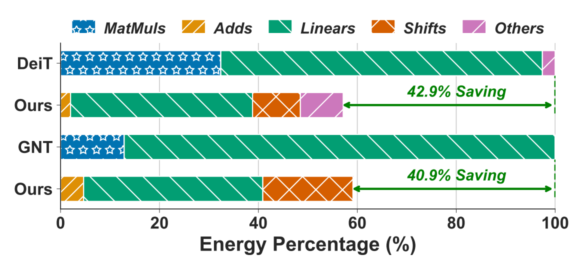


Figure 5: Energy breakdown on an Eyeriss accelerator.

Speedups of Shifts and Adds. Apart from the overall comparison and energy reduction profiling, we also test the GPU speedups of our customized `Shift` and `Add` kernels, as shown in Fig. 3 and 4, respectively. We can see that both customized kernels achieve faster speeds than PyTorch and

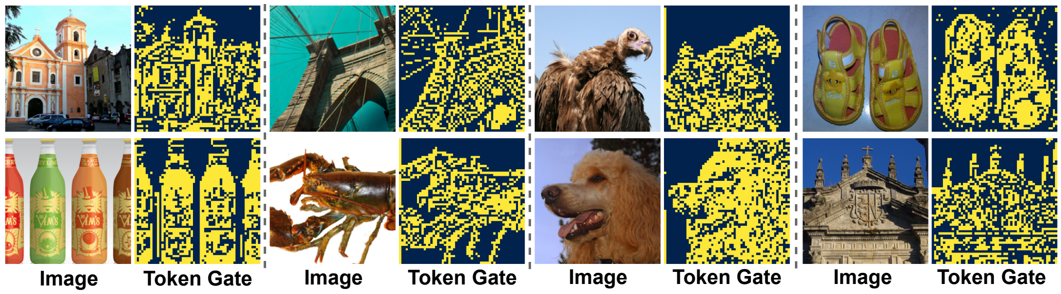


Figure 6: Visualization of the token dispatches in MoE routers/gates given images from the validation set, where yellow color denotes `Mult.` experts while blue color refers to `Shift` experts.

TVM baselines. For example, our `MatAdds` achieve on average **7.54**$\times$/**1.51**$\times$ speedups than PyTorch and TVM `MatMuls`, respectively. Our `MatShifts` achieve on average **2.35**$\times$/**3.07**$\times$/**1.16**$\times$ speedups than PyTorch `FakeShifts` [17], TVM `FakeShifts`, and TVM `MatMuls`, respectively. Note that in those comparisons, we use the default `einsum` operator for the PyTorch `MatMuls`, floating-point multiplication with power-of-twos for FakeShift, and implement the `MatAdds` and `MatShifts` using TVM [10] ourselves. We compare those customized operators with their multiplication counterparts in the same setting for fair comparisons. The speedup of `MatAdds` is because we replace multiplication-and-accumulation with accumulation only. The speedup of `MatShifts` is mainly attributed to the bit reductions (`INT32` and `INT8` for inputs and shift signs/weights) and thus data movement reductions instead of computations which are almost fully hidden behind data movements.

Ablation studies of Latency-aware Load-balancing Loss (LL-Loss). To demonstrate the effectiveness of the proposed LL-Loss in our ShiftAddViT MoE framework, we conduct ablation studies on two PVT models as shown in Tab. 7. We find that ShiftAddViT w/ LL-Loss achieves better accuracy-efficiency tradeoffs, e.g., **14.6**% latency reductions while maintaining comparable accuracies ($\pm$0.1%) when considering `Mult.` and `Shift` experts. The latency reduction could be larger if we consider more unbalanced experts with distinct runtimes. This set of experiments justifies the effectiveness of our LL-Loss in the MoE system.

Table 7: Ablation studies of the ShiftAddViT w/ or w/o LL-Loss on PVT models.

Models	Methods	Acc. (%)	Norm. Latency
PVTv2-B0	w/o LL-Loss	70.38	100%
	w/ LL-Loss	**70.37**	**85.4%**
PVTv1-T	w/o LL-Loss	74.73	100%
	w/ LL-Loss	**74.66**	**85.5%**

Visualization of Token Dispatches in MoE. To validate our hypothesis that important yet sensitive tokens require powerful experts while other insensitive tokens can be represented with much cheaper `Shift` experts, we visualize the token dispatches in MoE routers/gates of the first `MLP` layer in our ShiftAddViT PVTv2-B0 model as shown in Fig. 6. We see that our designed router successfully identifies the important tokens that contain object information, which are then dispatched to more powerful `Multiplication` experts, leaving unimportant tokens, e.g., background, to cheaper `Shift` tokens. This set of visualization further validates the hypothesis and explains why our proposed MoE framework can effectively boost our ShiftAddViT towards better accuracy-efficiency trade-offs.

5.5 Limitation and Societal Impact Discussion

We made a firm step to show the practical usage of the proposed hardware-inspired ShiftAddViT. While this relies on dedicated TVM optimization, the full potential can be unraveled with customized hardware accelerators toward naturally hardware-efficient ViTs. Also, the unbalanced MoE framework shows its generalizability but highly demands system support with ideal parallelism.

6 Conclusion

In this paper, we for the first time propose a hardware-inspired multiplication-reduced ViT model dubbed ShiftAddViT. It reparameterizes both attention and MLP layers in pre-trained ViTs with a mixture of multiplication primitives, e.g., bitwise shifts and adds, towards end-to-end speedups on GPUs and dedicated hardware accelerators without the need for training from scratch. Moreover, a novel mixture of unbalanced experts framework equipped with a new latency-aware load-balancing loss is proposed in pursuit of double-winning accuracy and hardware efficiency. We use multiplication or its primitives as experts in our ShiftAddViT cases. Extensive experiments on both 2D and 3D Transformer-based vision tasks consistently validate the superiority of our proposed ShiftAddViT as compared to multiple ViT baselines. We believe that this paper opens up a new perspective on designing energy-efficient ViT inference based on widely available pre-trained ViT models.

References

[1] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. Advances in neural information processing systems, 34:20014–20027, 2021.

[2] Moab Arar, Ariel Shamir, and Amit H Bermano. Learned queries for efficient local attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10841–10852, 2022.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461, 2022.

[4] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. arXiv preprint arXiv:2209.07484, 2022.

[5] Han Cai, Chuang Gan, and Song Han. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. arXiv preprint arXiv:2205.14756, 2022.

[6] 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.

[7] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. arXiv preprint arXiv:2103.14899, 2021.

[8] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 1468–1477, 2020.

[9] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 12270–12280, 2021.

[10] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. Tvm: An automated end-to-end optimizing compiler for deep learning. arXiv preprint arXiv:1802.04799, 2018.

[11] Xinghao Chen, Chang Xu, Minjing Dong, Chunjing Xu, and Yunhe Wang. An empirical study of adder neural networks for object detection. Advances in Neural Information Processing Systems, 34:6894–6905, 2021.

[12] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE journal of solid-state circuits, 52(1):127–138, 2016.

[13] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In International Conference on Learning Representations, 2021.

[14] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.

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

[17] Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2359–2368, 2021.

[18] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6824–6835, 2021.

[19] Hongxiang Fan, Thomas Chau, Stylianos I Venieris, Royson Lee, Alexandros Kouris, Wayne Luk, Nicholas D Lane, and Mohamed S Abdelfattah. Adaptable butterfly accelerator for attention-based nns via hardware and algorithm co-design. In 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 599–615. IEEE, 2022.

[20] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. The Journal of Machine Learning Research, 23(1):5232–5270, 2022.

[21] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. arXiv preprint arXiv:2104.01136, 2021.

[22] Bah-Hwee Gwee, Joseph S Chang, Yiqiong Shi, Chien-Chung Chua, and Kwen-Siong Chong. A low-voltage micropower asynchronous multiplier with shift–add multiplication approach. IEEE Transactions on Circuits and Systems I: Regular Papers, 56(7):1349–1359, 2008.

[23] Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. Fastmoe: A fast mixture-of-expert training system. arXiv preprint arXiv:2103.13262, 2021.

[24] Yefei He, Zhenyu Lou, Luoming Zhang, Weijia Wu, Bohan Zhuang, and Hong Zhou. Bivit: Extremely compressed binary vision transformer. arXiv preprint arXiv:2211.07091, 2022.

[25] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In International Conference on Computer Vision (ICCV), 2021.

[26] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research, 18(1):6869–6898, 2017.

[27] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 19–28, 2017.

[28] 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.

[29] Wenshuo Li, Xinghao Chen, Jinyu Bai, Xuefei Ning, and Yunhe Wang. Searching for energy-efficient hybrid adder-convolution neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1943–1952, 2022.

[30] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4804–4814, 2022.

[31] Xiaocong Lian, Zhenyu Liu, Zhourui Song, Jiwu Dai, Wei Zhou, and Xiangyang Ji. High-performance fpga-based cnn accelerator with block-floating-point arithmetic. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(8):1874–1885, 2019.

[32] Jing Liu, Zizheng Pan, Haoyu He, Jianfei Cai, and Bohan Zhuang. Ecoformer: Energy-saving attention with linear complexity. In NeurIPS, 2022.

[33] 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, pages 10012–10022, 2021.

[34] Zechun Liu, Barlas Oguz, Aasish Pappu, Lin Xiao, Scott Yih, Meng Li, Raghuraman Krishnamoorthi, and Yashar Mehdad. Bit: Robustly binarized multi-distilled transformer. arXiv preprint arXiv:2205.13016, 2022.

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2017.

[36] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: softmax-free transformer with linear complexity. Advances in Neural Information Processing Systems, 34:21297–21309, 2021.

[37] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178, 2021.

[38] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG), 38(4):1–14, 2019.

[39] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1):99–106, 2021.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32, 2019.

[41] Zheng Qu, Liu Liu, Fengbin Tu, Zhaodong Chen, Yufei Ding, and Yuan Xie. Dota: detect and omit weak attentions for scalable transformer acceleration. In Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pages 14–26, 2022.

[42] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in neural information processing systems, 34:13937–13949, 2021.

[43] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. Advances in Neural Information Processing Systems, 34:8583–8595, 2021.

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520, 2018.

[45] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538, 2017.

[46] Haichen Shen, Jared Roesch, Zhi Chen, Wei Chen, Yong Wu, Mu Li, Vin Sharma, Zachary Tatlock, and Yida Wang. Nimble: Efficiently compiling dynamic neural networks for model inference. Proceedings of Machine Learning and Systems, 3:208–222, 2021.

[47] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 3531–3539, 2021.

[48] Han Shu, Jiahao Wang, Hanting Chen, Lin Li, Yujiu Yang, and Yunhe Wang. Adder attention for vision transformer. Advances in Neural Information Processing Systems, 34:19899–19909, 2021.

[49] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In ICCV, 2017.

[50] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is attention all that nerf needs? In The Eleventh International Conference on Learning Representations, 2023.

[51] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 10347–10357. PMLR, 18–24 Jul 2021.

[52] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV, pages 459–479. Springer, 2022.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[54] Guangting Wang, Yucheng Zhao, Chuanxin Tang, Chong Luo, and Wenjun Zeng. When shift operation meets vision transformer: An extremely simple alternative to attention mechanism. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 2423–2430, 2022.

[55] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[56] 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 (ICCV), pages 568–578, October 2021.

[57] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media, 8(3):415–424, 2022.

[58] Yunhe Wang, Mingqiang Huang, Kai Han, Hanting Chen, Wei Zhang, Chunjing Xu, and Dacheng Tao. Addernet and its minimalist hardware design for energy-efficient artificial intelligence. arXiv preprint arXiv:2101.10015, 2021.

[59] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4):600–612, 2004.

[60] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9127–9135, 2018.

[61] 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.

[62] Yixing Xu, Chang Xu, Xinghao Chen, Wei Zhang, Chunjing XU, and Yunhe Wang. Kernel based progressive distillation for adder neural networks. In Advances in Neural Information Processing Systems, volume 33, pages 12322–12333. Curran Associates, Inc., 2020.

[63] Ping Xue and Bede Liu. Adaptive equalizer using finite-bit power-of-two quantizer. IEEE transactions on acoustics, speech, and signal processing, 34(6):1603–1611, 1986.

[64] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10809–10818, 2022.

[65] Haoran You, Xiaohan Chen, Yongan Zhang, Chaojian Li, Sicheng Li, Zihao Liu, Zhangyang Wang, and Yingyan Lin. Shiftaddnet: A hardware-inspired deep network. Advances in Neural Information Processing Systems, 33:2771–2783, 2020.

[66] Haoran You, Baopu Li, Shi Huihong, Yonggan Fu, and Yingyan Lin. Shiftaddnas: Hardware-inspired search for more accurate and efficient neural networks. In International Conference on Machine Learning, pages 25566–25580. PMLR, 2022.

[67] Haoran You, Zhanyi Sun, Huihong Shi, Zhongzhi Yu, Yang Zhao, Yongan Zhang, Chaojian Li, Baopu Li, and Yingyan Lin. Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design. In The 29th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2023), 2023.

[68] Haoran You, Yunyang Xiong, Xiaoliang Dai, Bichen Wu, Peizhao Zhang, Haoqi Fan, Peter Vajda, and Yingyan Lin. Castling-vit: Compressing self-attention via switching towards linear-angular attention during vision transformer inference. In The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2023), 2023.

[69] Zhongzhi Yu, Yonggan Fu, Sicheng Li, Chaojian Li, and Yingyan Lin. Mia-former: efficient and robust vision transformers via multi-grained input-adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 8962–8970, 2022.

[70] 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.

[71] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 586–595, 2018.

[72] Yang Zhao, Chaojian Li, Yue Wang, Pengfei Xu, Yongan Zhang, and Yingyan Lin. Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures. In International Conference on Acoustics, Speech, and Signal Processing, pages 1593–1597, 05 2020.

[73] Zhe Zhou, Junlin Liu, Zhenyu Gu, and Guangyu Sun. Energon: Toward efficient acceleration of transformers using dynamic sparse attention. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(1):136–149, 2022.

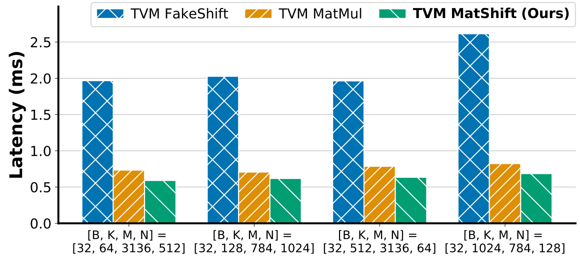


Figure 7: MLP/Linear latency speedups using shifts, where inputs are of shape $B \times K \times M$, weights are of shape $K \times N$. All dimensions are set w.r.t. the real dimensions in PVTs [56].

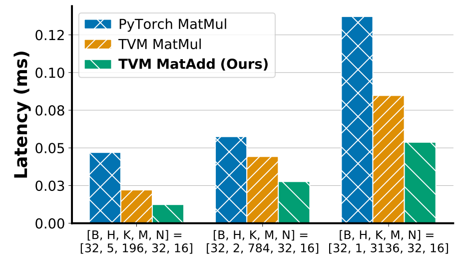


Figure 8: `MatMuls` latency speedups using adds, where inputs are of shape $B \times H \times K \times M$, weights are of shape $B \times H \times K \times N$.

A More Speedup Comparisons of `Shift` and `Add`

Apart from the speedup analysis when the batch size is one, we also conducted GPU speedup tests on our customized `Shift` and `Add` kernels when using larger batch sizes, such as 32, as depicted in Fig. 7 and 8, respectively. The results demonstrate that both our customized kernels consistently outperform the PyTorch and TVM baselines in terms of speed. For instance, our `MatAdds` achieve an average speedup of **2.80**$\times$/**1.65**$\times$ compared to PyTorch and TVM `MatMuls`, respectively. Similarly, our `MatShifts` achieve an average speedup of **3.38**$\times$/**1.21**$\times$ compared to TVM `FakeShifts` [17] and TVM `MatMuls`, respectively. It is important to note that in these comparisons, we ensured fair conditions by using the default `einsum` operator for PyTorch `MatMuls`, employing floating-point multiplication with power-of-twos for FakeShift, and implementing the `MatAdds` and `MatShifts` ourselves using TVM [10]. We compared these customized operators with their multiplication counterparts in the same setup to ensure a fair evaluation. The observed speedup of `MatAdds` is due to the replacement of multiplication-and-accumulation with accumulation only. On the other hand, the speedup of `MatShifts` is primarily attributed to the reduction in bit precision (`INT32` and `INT8` for inputs and shift signs/weights, respectively), resulting in reduced data movement instead of computations. These computational improvements are almost entirely hidden behind data movements.

B More Visualization of ShiftAddViT MoE Framework

In order to validate our hypothesis that important yet sensitive tokens require powerful experts, and other insensitive tokens can be represented by cheaper `Shift` experts, we have conducted additional token dispatch visualizations in the MoE routers/gates of the first `MLP` layer in our ShiftAddViT PVTv2-B0 model, as illustrated in Fig. 9. These visualizations provide further evidence that our designed routers successfully identify the crucial tokens that contain object information. Consequently, these tokens are dispatched to more powerful `Mult.` experts, while less important tokens, such as background tokens, are assigned to cheaper `Shift` experts. This collection of visualizations further reinforces our hypothesis and elucidates why our proposed MoE framework effectively enhances the accuracy-efficiency trade-offs of our ShiftAddViT model.

C More Results on 3D Tasks

We have extended our methods to 3D NVS tasks and have built ShiftAddViT on top of Transformer-based GNT models [50]. In the main paper, we tested and reported the averaged PSNR, SSIM, LPIPS, as well as latency and energy of both ShiftAddViT and baselines, i.e., the vanilla NeRF [39] and GNT [50], on the LLFF dataset across eight scenes. Here, we further supply the full results of eight scenes. As shown in Tab. 8, 9, 10, our ShiftAddViT consistently leads to better accuracy-efficiency tradeoffs in terms of the three metrics mentioned above, achieving comparable or even better generation quality ($\uparrow$0.55/$\downarrow$0.19 averaged PSNR across eight scenes), as compared to NeRF and GNT baselines, respectively. In particular, ShiftAddViT achieves better generation quality (up to $\uparrow$0.88 PSNR) than GNT on some representation scenes, e.g., Orchid and Flower. In all tables, the first, second, and third ranking performances are noted with corresponding colors.

In addition, we supply more qualitative rendering visualization on three scenes, Horns, Orchids, and Flowers, as shown in Fig. 10. We see that our ShiftAddViT generates more clear images than vanilla

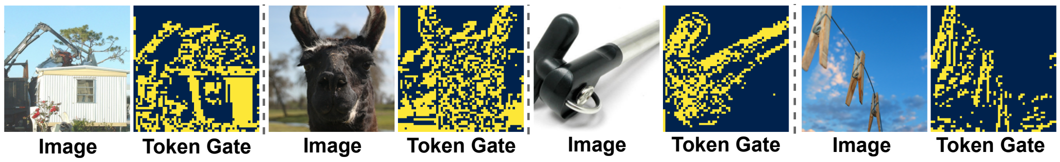


Figure 9: Visualization of the token dispatches in MoE routers/gates given images from the validation set, where yellow color denotes `Multiplication` experts and blue color refers to `Shift` experts.

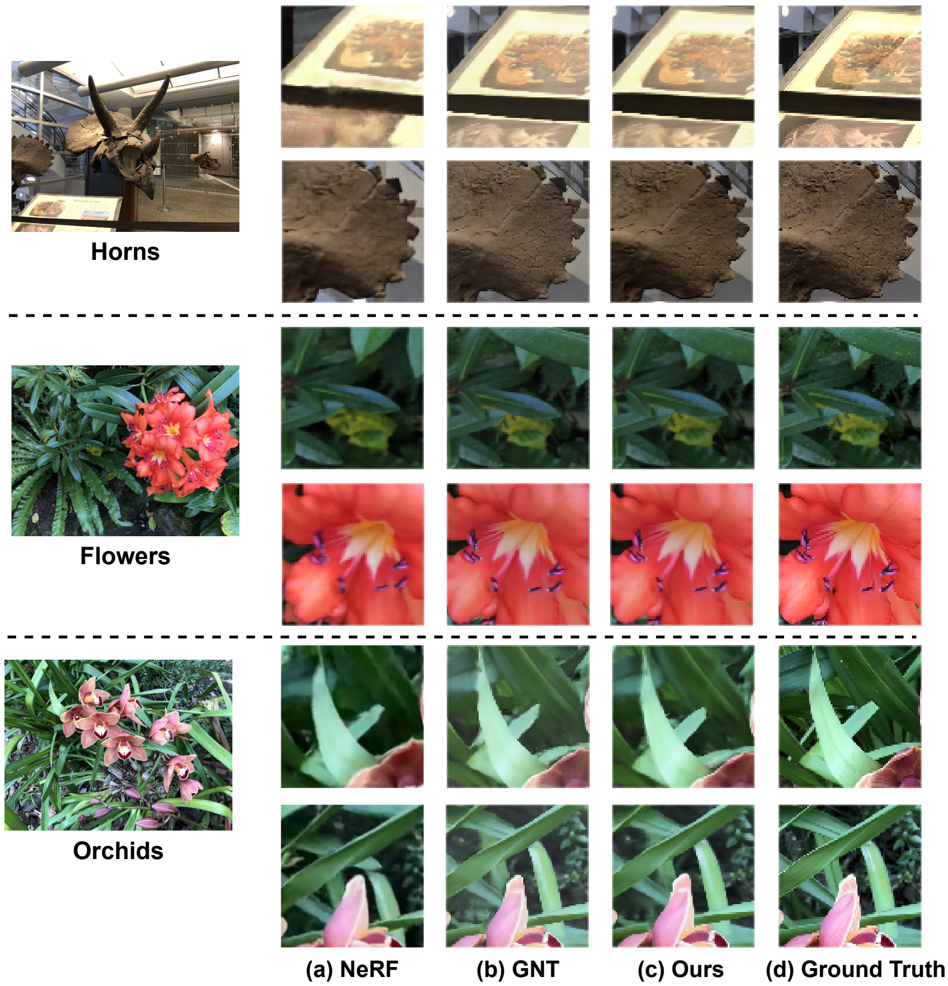


Figure 10: Qualitative comparisons between (a) NeRF [39], (b) GNT [50], (c) ShiftAddViT (Ours), and (d) ground truth when tested on the LLFF dataset for single-scene rendering.

NeRF [39] and comparable or even better details than GNT [50]. Quantitative and qualitative results consistently show that our ShiftAddViT does not hurt rendering quality when improving rendering speeds and energy efficiency.

D More Experiment Settings

For the classification task, we first replace the original attention layers in PVTv1 [56] and PVTv2 [57] with the standard MSA, and then finetune the MSA-based variants on top of the ImageNet-1k-pretrained weights for 100 epochs with a learning rate (lr) of 5×10^{-5} following Ecoformer [32]. After that, we reparameterize the MSA-based models through a two-stage finetuning process: (1) convert MSA to linear attention [68] and reparameterize `MatMuls` with add layers via 100 epochs of finetuning with a base lr of 1×10^{-5}, and (2) reparameterize MLPs or linear layers with shift or MoE

Table 8: Comparisons between ShiftAddViT and baselines. We apply our Shift&Add techniques on top of the SOTA Transformer-based NeRF model, GNT [50], and report averaged PSNR (↑) on the LLFF dataset [38] across eight scenes.

Methods	Add	Shift	MoE	PSNR (↑) on LLFF Dataset							
				Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns
NeRF [39]	-	-	-	32.70	25.17	20.92	31.16	20.36	27.40	26.80	27.45
GNT [50]	-	-	-	32.96	24.31	22.57	32.28	20.67	27.32	28.15	29.62
ShiftAddViT	✓	✗	✗	31.68	24.63	22.34	31.79	20.74	28.02	26.99	28.62
	✓	✓(Both)	✗	31.64	24.63	22.34	31.73	20.78	28.05	27.03	28.61
	✓	✓(Attn)	✓(MLPs)	31.85	24.65	22.37	31.85	20.73	28.20	27.08	28.62
	✗	✓(Both)	✗	32.12	24.35	22.42	32.15	20.84	28.14	27.32	29.09

Table 9: Comparisons between ShiftAddViT and baselines. We apply our Shift&Add techniques on top of the SOTA Transformer-based NeRF model, GNT [50], and report averaged SSIM (↑) on the LLFF dataset [38] across eight scenes.

Methods	Add	Shift	MoE	SSIM (↑) on LLFF Dataset							
				Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns
NeRF [39]	-	-	-	0.948	0.792	0.690	0.881	0.641	0.827	0.880	0.828
GNT [50]	-	-	-	0.963	0.846	0.852	0.934	0.752	0.893	0.936	0.935
ShiftAddViT	✓	✗	✗	0.949	0.839	0.838	0.922	0.730	0.891	0.916	0.911
	✓	✓(Both)	✗	0.948	0.839	0.839	0.923	0.730	0.892	0.917	0.911
	✓	✓(Attn)	✓(MLPs)	0.950	0.840	0.841	0.923	0.731	0.894	0.917	0.912
	✗	✓(Both)	✗	0.955	0.836	0.841	0.930	0.746	0.896	0.923	0.922

Table 10: Comparisons between ShiftAddViT and baselines. We apply our Shift&Add techniques on top of the SOTA Transformer-based NeRF model, GNT [50], and report averaged LPIPS (↓) on the LLFF dataset [38] across eight scenes.

Methods	Add	Shift	MoE	LPIPS (↓) on LLFF Dataset							
				Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns
NeRF [39]	-	-	-	0.178	0.280	0.316	0.171	0.321	0.219	0.249	0.268
GNT [50]	-	-	-	0.060	0.116	0.109	0.061	0.153	0.092	0.080	0.076
ShiftAddViT	✓	✗	✗	0.087	0.141	0.129	0.080	0.182	0.089	0.107	0.110
	✓	✓(Both)	✗	0.088	0.142	0.128	0.080	0.182	0.088	0.106	0.110
	✓	✓(Attn)	✓(MLPs)	0.086	0.140	0.127	0.079	0.180	0.087	0.104	0.109
	✗	✓(Both)	✗	0.076	0.138	0.125	0.070	0.169	0.083	0.097	0.097

layers through another 100 epochs of finetuning with a base lr of 1×10^{-5}. All models are trained using eight RTX A5000 GPUs with a total batch size of 256, and we use the AdamW optimizer [35] with a cosine decay lr scheduler. All other hyperparameters are the same as those in Ecoformer [32] and PVTv2 [57]. *For the NVS task,* we apply reparameterization on top of the pretrained models from GNT [50] using a similar two-stage fine-tuning process, except that we do not convert MSA to linear attention for maintaining accuracy. Specifically, we first (1) reparameterize `MatMuls` with add layers and then (2) reparameterize MLPs or linear layers with shift or MoE layers. Both stages are finetuned for 140K steps with a base lr of 5×10^{-4}, and we sample 2048 rays with 192 coarse points sampled per ray in each iteration. All other hyperparameters are the same as those in GNT [50], including the use of the Adam optimizer with an exponential decay lr scheduler.