
CASCADEViT: CASCADED CHUNK-FEEDFORWARD AND CASCADED GROUP ATTENTION VISION TRANSFORMER

Srivathsan Sivakumar

Ontario Tech University, Canada
 srivathsan.sivakumar@ontariotechu.net

Faisal Z. Qureshi

Ontario Tech University, Canada
 Faisal.Qureshi@ontariotechu.ca

ABSTRACT

Vision Transformers (ViTs) have demonstrated remarkable performance across a range of computer vision tasks; however, their high computational, memory, and energy demands hinder deployment on resource-constrained platforms. In this paper, we propose *Cascaded-ViT (CViT)*, a lightweight and compute-efficient vision transformer architecture featuring a novel feedforward network design called *Cascaded-Chunk Feed Forward Network (CCFFN)*. By splitting input features, CCFFN improves parameter and FLOP efficiency without sacrificing accuracy. Experiments on ImageNet-1K show that our *CViT-XL* model achieves 75.5% Top-1 accuracy while reducing FLOPs by 15% and energy consumption by 3.3% compared to EfficientViT-M5. Across various model sizes, the CViT family consistently exhibits the lowest energy consumption, making it suitable for deployment on battery-constrained devices such as mobile phones and drones. Furthermore, when evaluated using a new metric called *Accuracy-Per-FLOP (APF)*, which quantifies compute efficiency relative to accuracy, CViT models consistently achieve top-ranking efficiency. Particularly, CViT-L is 2.2% more accurate than EfficientViT-M2 while having comparable APF scores.

1 Introduction

Vision Transformers (ViTs) are large, sophisticated, and data-hungry models that provide exceptional performance and modeling capacity [1, 2, 3, 4, 5]. CoCa [6], a 2.1 billion-parameter image-text encoder-decoder foundation model, achieved state-of-the-art 91% Top-1 classification accuracy on ImageNet-1K [7] using a fine-tuned encoder. In comparison, the best-performing convolutional neural network (CNN) model [8] achieves 90% Top-1 classification accuracy on the same dataset.

Due to their groundbreaking performance, Vision Transformers (ViTs) are increasingly being deployed in real-time applications and battery-constrained devices such as Unmanned Aerial Vehicles (UAVs) [9, 10, 11] and wearable technologies [12]. However, models like CoCa [6] have substantial computational and memory requirements, making their deployment in resource-limited environments a significant challenge [13]. This highlights the urgent need for the design of lightweight and compute-efficient ViT architectures that can operate effectively under stringent hardware and energy constraints.

Several lightweight Vision Transformer (ViT) architectures have been proposed to reduce model size and complexity for image classification tasks, including MiniViT [14], EdgeViT [15], and TinyViT [16]. While these models are considerably smaller than standard ViTs, they still require significantly more computational resources than EfficientViT [5]. As shown in Table 2, the largest EfficientViT variant achieves approximately four times higher throughput than the smallest MiniViT model (Mini-DeiT-Ti). However, most efficient ViT designs primarily optimize for latency, parameter count, and FLOPs, often overlooking energy consumption [5, 14, 16, 17] which is a critical factor for deployment in battery-powered systems.

EfficientViT achieves its efficiency by replacing self-attention modules with feedforward networks (FFNs), which have been shown to exhibit redundancy [18]. To mitigate this redundancy and further improve efficiency, techniques such as architecture-adaptive optimization (e.g., weight sharing) and modular designs like Chunk-FFNs (CFFN) [19] have been proposed. These strategies not only reduce memory and energy consumption but may also enhance model accuracy [20].

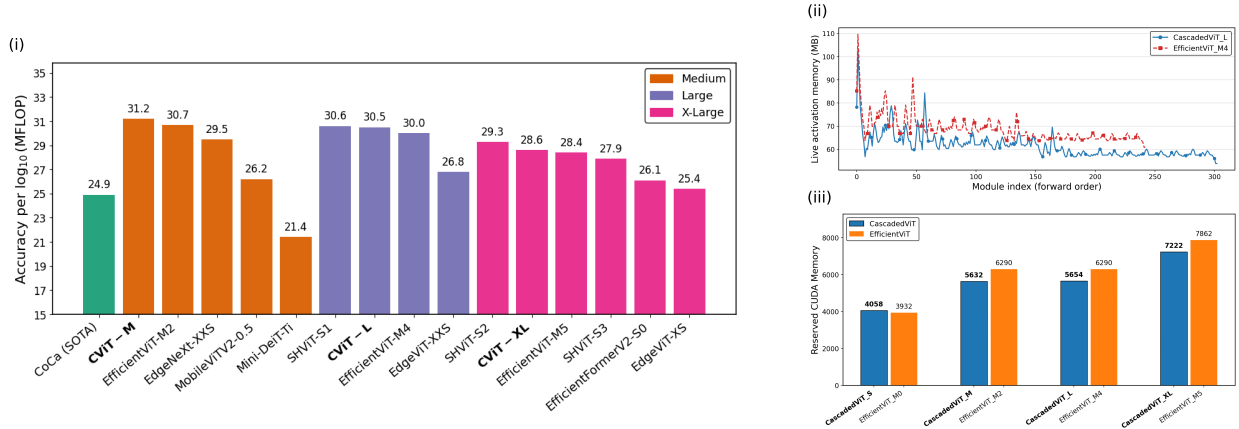


Figure 1: (i) Accuracy per \log_{10} MFLOP (APF) for all evaluated models. Models are grouped by size, and higher APF indicates greater Top-1 accuracy per unit of computation. (ii) Live memory trace of CViT-L and EfficientViT-M4. CViT-L consistently demands less memory indicating system-wide memory efficiency. (iii) Reserved memory (MB) demands of CViT and EfficientViT. CViT demands less memory allocation across all scales except small.

In this paper, we introduce **Cascaded-ViT (CViT)**, a ViT variant optimized for battery-powered edge deployment, that addresses the dominant inference bottleneck—the FFNs [21]—in EfficientViT by replacing FFNs with Cascaded-Chunk Feed Forward Network (CCFFN). The CCFFN module partitions the input feature map along the channel dimension into smaller subsets and applies a lightweight FFN independently to each subset. The outputs from earlier FFNs are progressively aggregated through residual connections, enabling deeper feature refinement with minimal overhead. Throughout this work, we use the terms “EfficientViT” and “backbone” interchangeably to refer to the underlying network framework.

We evaluate CViT against several state-of-the-art efficient and edge-optimized ViT architectures. Experimental results on ImageNet-1K [7] show that CViT achieves competitive accuracy while substantially reducing computational cost, both in terms of energy consumption and memory footprint. Specifically, **CViT-M** reduces FLOPs by **15%** and parameters by **0.7M** compared to EfficientViT-M2, with only a **0.9%** drop in Top-1 accuracy. Additionally, CViT yields a smaller memory footprint and lower per-image energy than competing methods. To better capture the trade-off between accuracy and computational budget, we introduce a new metric—**Accuracy per FLOP (APF)** which quantifies compute efficiency relative to predictive performance. CViT achieves a favorable position on this APF curve, establishing a new Pareto frontier for real-world ViT deployment under resource constraints.

2 Related Works

As discussed earlier, Vision Transformer (ViT) models are large and data-intensive, which limits their deployment in resource-constrained environments. Consequently, there is strong motivation to compress these models to enable their use in edge devices and energy-limited systems. Two widely adopted compression strategies include: (1) pruning and (2) weight sharing. However, we argue that efficient architectural design remains the most effective means of achieving ViT efficiency. As demonstrated in Table 7, compression techniques such as clustered weight sharing often yield limited practical benefits.

Pruning techniques aim to reduce inference latency by removing redundant parameters and optimizing matrix operations [22]. For example, [23] proposed a Hessian-based pruning strategy that facilitates more effective parameter redistribution. [24] introduced a Taylor-based approximation method to identify strong component matches for collaborative pruning. Token- and feature-level pruning [25, 26, 27, 28] is another common technique that reduces ViT computation by eliminating less informative regions of the input. [29] proposed an edge- and energy-aware dynamic pruning method that drops tokens with low attention scores, while retaining those with high scores. [30] further demonstrated that dynamic reconfiguration of transformers at runtime, based on hardware and software profiles, can extend battery life by up to 4x.

Weight sharing—first introduced in the 1980s [31]—ties parameters across layers to reduce model size and increase efficiency [20, 32, 33]. While this technique has shown promise in improving parameter efficiency, it can also reduce model flexibility and accuracy if not carefully optimized [20]. Recent work has focused on mitigating this trade-

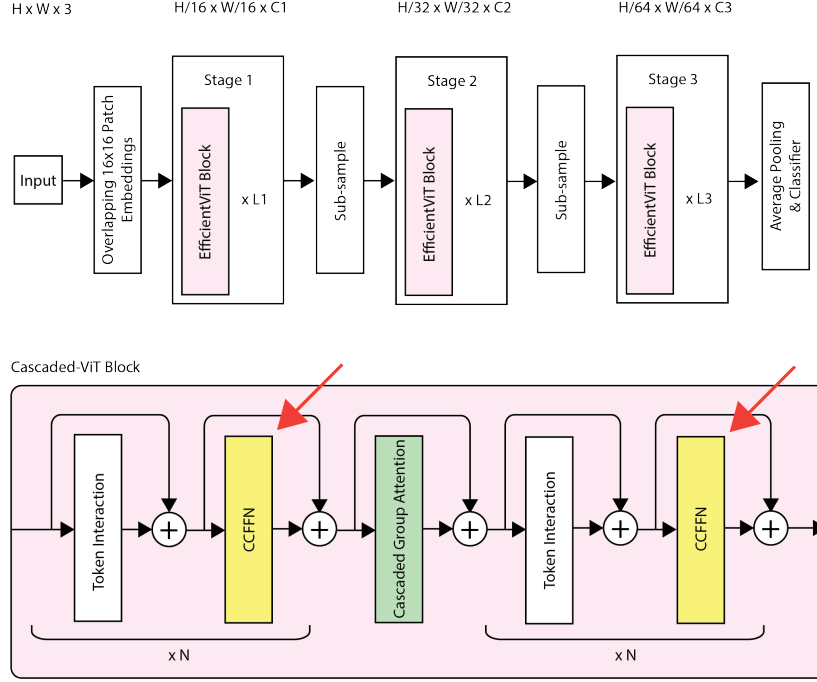


Figure 2: An overview of the EfficientViT (top). The proposed architecture replaces EfficientViT Block with the Cascaded Chunk ViT Block (bottom). Cascaded Chunk ViT Block replaces the pre- and post-attention FFNs in the EfficientViT Block with Cascaded Chunk FFNs (shown in Yellow and highlighted with Red arrows).

off [34, 35]. MiniViT [14], for instance, introduced weight multiplexing, combining shared weights with transformations to simulate demultiplexing. However, the added transformations increase computational overhead. [36] proposed residual weight sharing, where each shared layer includes a residual component, increasing model capacity with only a modest increase in parameter count.

Efficient architectural design is another promising approach and often involves hybrid CNN-ViT structures [37, 38, 39]. MobileViTv3 [40] replaces local convolutional operations with global transformer-based computations. Its smallest variant, MobileViTv3-XXS, achieves 70.98% top-1 accuracy on ImageNet-1K [7] with only 1.2 million parameters. EfficientFormer [17], a lightweight transformer architecture, employs latency-aware pruning and maintains consistent tensor dimensions. EfficientFormer-L1 achieves an inference latency of just 1.6 ms on an iPhone 12. Recently, MicroViT [41] introduced an energy-efficient Single Head Attention mechanism that processes only one-quarter of the input features in the attention block, significantly reducing model size and power consumption.

3 The Making of Cascaded-ViT

Cascaded-ViT replaces the Feed-Forward Network (FFN) layers in EfficientViT with Cascaded Chunked Feed-Forward Network (CCFFN) layers [5] (Figure 2). We begin by reviewing the architecture of EfficientViT. Figure 2 provides an overview.

At a high level, EfficientViT consists of three stages, each containing a fixed number of EfficientViT blocks. Each block comprises five components: Token Interaction, Pre-Attention Feed-Forward Network, Cascaded Group Attention (CGA), Token Interaction, and Post-Attention Feed-Forward Network. The input to the first stage is generated by applying overlapping 16×16 patch embeddings to the input image. Between the stages, subsampling layers reduce the spatial resolution by half. Finally, the output of the last stage is processed via global average pooling and passed to a classifier for prediction.

EfficientViT’s efficiency stems from three key architectural innovations:

Model	Depth	Emb. dim.	# Heads
CViT-S	[1, 2, 3]	[64, 128, 192]	[2, 3, 3]
CViT-M	[1, 2, 3]	[128, 192, 224]	[4, 3, 2]
CViT-L	[1, 2, 3]	[128, 256, 384]	[4, 4, 4]
CViT-XL	[1, 3, 4]	[192, 288, 384]	[3, 3, 4]

Table 1: Cascaded-ViT is available in multiple model sizes. In this configuration, *Depth* refers to the number of Cascaded-ViT blocks in each stage, *Embedding Dimension* indicates the number of feature channels after each stage, and *Heads* denotes the number of attention heads used in the Cascaded Group Attention module.

1. **Memory efficiency:** Memory-intensive self-attention layers are replaced with lightweight feed-forward networks (FFNs). A single self-attention module is positioned between pre-attention and post-attention FFNs, significantly reducing memory usage.
2. **Computational efficiency:** To lower computational cost, only a subset of input features is routed to each attention head. The outputs of successive heads are accumulated and added to the next subset of the input feature, improving performance while reducing FLOPs and parameters.
3. **Parameter efficiency:** Parameters are strategically reallocated—more to critical components and fewer to less influential ones. Given the high redundancy in FFNs [18], their expansion ratios are halved, leading to a leaner yet effective model.

EfficientViT is particularly well-suited for deployment to edge devices. For instance, EfficientViT-M0 contains only 2.3 million parameters, achieves 63.2% top-1 accuracy on ImageNet-1K [7], and can run inference smoothly on an iPhone 11 [5].

3.1 Beyond EfficientViT

This work began as an investigation into how to make EfficientViT even more “efficient.” On one hand, we drew inspiration from model compression techniques such as weight sharing and pruning, which are widely studied for reducing model size and computational cost [20, 42]. On the other hand, we followed principled approaches to efficient network design based on low-complexity Vision Transformers and chunk-level FFN optimization [19, 41].

While both directions were explored, we found that replacing the pre- and post-attention FFN layers in EfficientViT blocks with *Cascaded Chunk Feed-Forward Networks (CCFFNs)* led to substantial improvements. Specifically, this modification significantly reduces model complexity in terms of parameter count and FLOPs, while maintaining competitive performance across standard vision tasks such as classification, object detection, and semantic segmentation. (Results related to weight sharing are included in the ablation study in Section 4.6). Cascaded-ViT uses *Batch Normalization* (BatchNorm) [43] throughout the backbone, as it can be fused with convolutional layers to improve inference speed. The activation function used is *ReLU* [44], chosen for its computational efficiency and broad hardware compatibility. Similar to EfficientViT, in each subsampling block, the Cascaded Group Attention (CGA) module is replaced by an *inverted residual block*, following the design principles proposed in [45, 46]. Each subsampling block also incorporates a *Squeeze-and-Excite (SE)* module [47] to enhance feature recalibration and emphasize informative channels, while reducing the spatial resolution by a factor of two.

3.2 Chunk FFN

Originally proposed for a speech recognition Transformer model [19], the Chunk Feed Forward Network (CFFN) is an efficient compression technique in which the input is partitioned into smaller chunks, each processed by a lightweight feed-forward network (FFN). Following the formulation in [48], the FFN is redefined as a key-value mechanism and expressed as:

$$\text{FFN}(\mathbf{X}) = f(\mathbf{X}\mathbf{K}^T)\mathbf{V} \quad (1)$$

where \mathbf{K} and \mathbf{V} denote learnable weight matrices, and f represents a non-linear activation function. The Chunk-FFN effectively captures diverse feature representations while leveraging the key-value memory structure in Eq. 1, without incurring high computational costs. This design leads to a substantial reduction in parameters while maintaining stable

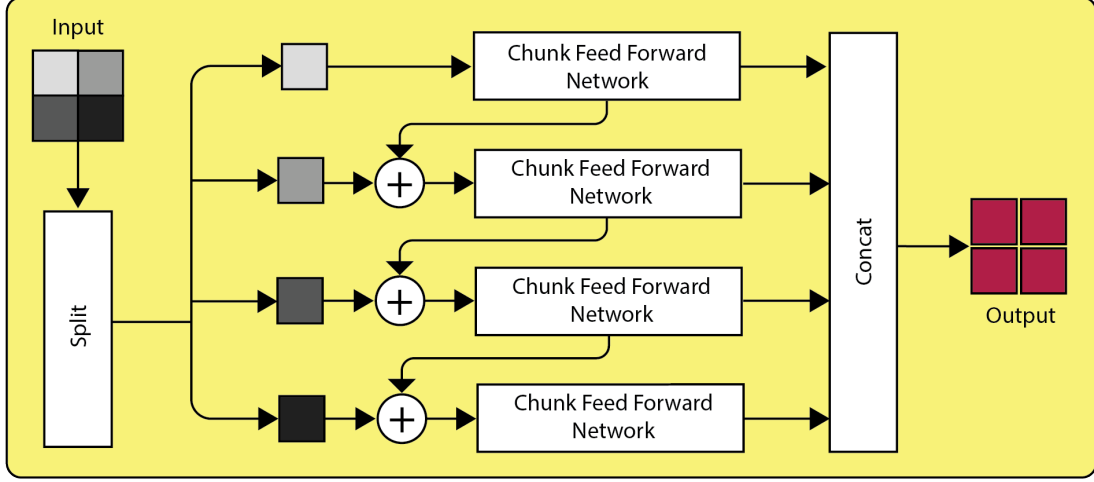


Figure 3: Cascaded-Chunk FFN layer that replaces the FFN layers in EfficientViT Blocks.

performance. Empirically, the model demonstrated improved results on Aishell-1 [49] and HKUST [50], achieving 0.3% and 0.2% reductions in Character Error Rate (CER), respectively.

3.3 Cascaded-Chunk FFN

Although the expansion ratio of FFN layers in the backbone is reduced from 4 to 2 as in [5], we argue that these layers can be further optimized to create a smaller and more efficient model. To this end, inspired by CFFN [19] and the Cascaded Group Attention (CGA) mechanism [5] for Vision Transformers, we introduce Cascaded-Chunk FFN (CCFFN).

As illustrated in Figure 3, the input feature is divided into multiple chunks, each passed through a compact FFN. The output from one FFN is added to the next chunk and then used as the input to the subsequent FFN, forming a cascade of progressively enriched features. Drawing from [19, 5], the CCFFN module can be formally expressed as:

$$\mathbf{X}_1, \dots, \mathbf{X}_n = \text{Split}(\mathbf{X}), \quad (2)$$

$$\mathbf{X}'_i = \begin{cases} \mathbf{X}_i, & \text{if } i = 1 \\ \mathbf{X}_i + \mathbf{Y}_{i-1}, & \text{if } i > 1 \end{cases} \quad (3)$$

$$\mathbf{Y}_i = \text{FFN}_i(\mathbf{X}'_i), \quad (4)$$

$$\text{Output} = \text{Concat}(\mathbf{Y}_1, \dots, \mathbf{Y}_n), \quad (5)$$

where $1 \leq i \leq n$ and n is the number of chunks. \mathbf{X}_i is the i^{th} subset of the input. \mathbf{X}'_i is the input to the FFN. \mathbf{Y}_i is the i^{th} output from the CFFN. We further boost the capacity of the model by computing the feature maps of each CFFN in a cascaded manner as defined in Eq 3. The output from each CFFN is added to the next subset of input which promotes the learning of more refined features. \mathbf{X}'_i is the sum of the i^{th} subset of the input \mathbf{X}_i and the $(i-1)^{\text{th}}$ output from the CFFN. When $i = 1$, the input remains unaffected as there are no CFFNs before the first one.

This new CCFFN module saves approximately 20% parameters and 15% FLOPs compared to the backbone. Crucially, CCFFN posts lower live and reserved memory demands, which is highly desirable for deployment in resource-constrained edge devices [51, 52, 53]. Furthermore, The cascading nature of the module increases the depth of the model, potentially boosting its capacity while not adding to the parameter count. Since we are launching more FFNs compared to the backbone, a slight but noticeable increase in latency is expected.

4 Experiments

The training recipe used by [5] introduces instability, likely due to aggressive image augmentations being incompatible with FFNs that operate on reduced channel dimensions. As a result, careful hyperparameter tuning is necessary to

stabilize training under the modified architecture. We train CViT from scratch on ImageNet-1K [7] using a single NVIDIA GH200-96GB GPU for 300 epochs, employing the AdamW optimizer [54]. The weight decay is set to 1.25×10^{-2} , with a maximum learning rate of 9×10^{-4} and a minimum of 1×10^{-4} . A batch size of 3072 is used to further reinforce training stability. Mixup and CutMix ratios are set to 0.6 and 0.8, respectively.

We adopt the same augmentation techniques and single-cycle cosine annealing schedule as in [5, 55], which include AutoAugment [56], Mixup [57], and Random Erasing [58].

To evaluate computational performance, we collect throughput metrics across different platforms using a batch size of 2048. GPU inference is measured on an NVIDIA V100-SXM2-32GB, Apple Silicon performance on an Apple M4 Pro chip, and NPU inference on an AMD Ryzen AI chip. For Ryzen AI, models are quantized to 8-bit integers and converted to ONNX format for compatibility. CPU throughput is measured on an Intel Xeon Silver 4114 @ 2.20GHz with batch size 16, using a single thread in accordance with [39].

Energy consumption during inference is measured on a MacBook Pro with the Apple M4 Pro chip. GPU power readings are obtained via powermetrics at 1 Hz sampling frequency, which reports two GPU readings per second. The idle GPU power range is identified as $(8 \pm 1) \times 10^{-3}$ W. During inference, peak GPU power values are recorded and averaged across approximately 133 readings per model. With each model running for $133/2 = 66.5$ seconds on average, total energy consumption is computed as the product of average power usage and runtime duration.

4.1 Knowledge Distillation

We train CViT-XL and CViT-L from scratch on ImageNet-1K [7] using the training recipe described previously, and employ them as teacher models in a knowledge distillation (KD) framework to enhance the performance of smaller models. During student training, we reduce the CutMix and Mixup ratios to 0.1, set the distillation loss weight α to 0.5, and use a temperature of 2.0. Additionally, we lower the minimum learning rate to 9×10^{-5} to facilitate finer weight updates during later training stages.

Interestingly, using CViT-XL as the teacher for CViT-S and CViT-M led to a performance drop, even when experimenting with higher temperature values to soften the teacher’s outputs. This degradation is likely due to the teacher’s predictions being overly confident, which small-capacity student models struggle to learn from effectively [59]. Consequently, we selected CViT-L as the teacher for CViT-S and CViT-M, and used CViT-XL to distill CViT-L, keeping the same KD hyperparameters across all configurations.

The results show consistent gains: CViT-S, CViT-M, and CViT-L improve their Top-1 accuracy by 2%, 1.4%, and 0.6%, respectively, when trained with their corresponding teacher models.

4.2 ImageNet-1K

We compare our CViT models against several small and lightweight Vision Transformers, including EfficientViT [5], MiniViT [14], TinyViT [16], SHViT [60], and EdgeViT [15], with results summarized in Table 2. CViT consistently demonstrates the lowest or near-lowest computational and energy requirements while maintaining competitive Top-1 accuracy. The integration of the CCFFN module enables our models to remain among the smallest within each size category.

While some models such as EfficientFormerV2-S0 [17] contain fewer parameters, they incur substantially higher FLOPs (e.g., 800 MFLOPs). In contrast CViT achieves both parameter and compute efficiency, offering a 20% reduction in parameter count and a 15% reduction in FLOPs compared to the baseline backbone across all model sizes. CViT-S is particularly efficient, requiring the least FLOPs and energy among its peers, with only a 1.3% Top-1 accuracy trade-off compared to EfficientViT-M0 [5]. Compared to MobileViTv2-0.5 [61], CViT-M uses $2.8\times$ less compute while maintaining similar accuracy. TinyViT-5M [16] achieves 9.2% higher accuracy than CViT-M but at a cost of $15\times$ more FLOPs.

SHViT-S1 [60] and CViT-L are comparable in accuracy, with CViT-L achieving a 0.2% higher Top-1 score at the expense of just 8 additional MFLOPs. Notably, SHViT-S1 consumes $1.25\times$ more energy during inference, indicating that CViT-L, despite slightly higher compute, is more energy-efficient. CViT-L, while two sizes larger than EfficientViT-M2, demands similar energy consumption while being 2.2% more accurate. A similar trend is observed between SHViT-S2 and CViT-XL, where CViT-XL achieves 0.3% higher accuracy, has 1.6M fewer parameters, only 16% more FLOPs and consumes nearly 120mJ/img less energy during inference. CViT-XL exhibits FLOP counts that are orders of magnitude lower than models like TinyViT [16], MiniViT [14], and EdgeViT [15], despite incurring only a 5.4%–5.8% accuracy trade-off. Across all sizes, CViT requires an average of 5% less energy compared to EfficientViT.

Model	Top-1 (%)	Top-5 (%)	FLOPs (M)	Params (M)	Input	Epochs	Throughput (images/s)				Energy/img (mJ)
							GPU	CPU	M4 Pro	RyzenAI	
CViT-S	62.0	84.2	67	1.9	224	300	25740	107	5775	1453	471 ± 29
EfficientViT-M0 [5]	63.2	85.4	79	2.3	224	300	30692	107	5931	1590	490 ± 23
EdgeNeXt-XXS [62]	71.2	-	261	1.3	256	300	-	32	930	-	<i>1166 ± 35</i>
MobileViTV2-0.5 [61]	70.2	-	480	1.4	256	300	4403	9	24	17	-
Mini-DeiT-Ti [14]	73.0	91.6	2600	3.0	224	300	3145	4	731	96	714 ± 39
CViT-M	69.9	89.5	173	3.5	224	300	20464	58	3717	867	568 ± 52
EfficientViT-M2 [5]	70.8	90.2	201	4.2	224	300	21432	58	3738	911	581 ± 36
EdgeViT-XXS [15]	74.4	-	600	4.1	224	300	4513	15	1056	119	665 ± 19
EfficientFormerV2-S0 [17]	75.7	-	800	3.5	224	300	1180	16	44	185	<i>1286 ± 32</i>
TinyViT-5M [16]	79.1	94.8	2600	5.4	224	300	3148	6	96	-	<i>1170 ± 87</i>
SHViT-S1 [60]	72.8	91.0	241	6.3	224	300	23132	63	4513	1168	737 ± 16
CViT-L	73.0	91.2	249	7.0	224	300	17335	45	2978	667	588 ± 42
EfficientViT-M4 [5]	74.3	91.8	299	8.8	224	300	18132	42	2894	742	620 ± 45
EdgeViT-XS [15]	77.2	-	1100	6.7	224	300	3502	9	472	28	<i>1363 ± 64</i>
SHViT-S2 [60]	75.2	92.4	366	11.4	224	300	7738	44	2903	904	783 ± 16
CViT-XL	75.5	92.4	435	9.8	224	300	11934	27	1910	423	653 ± 16
EfficientViT-M5 [5]	77.1	93.4	522	12.4	224	300	12098	26	1900	556	675 ± 23
EdgeViT-S [15]	81.0	-	1900	11.1	224	300	2455	6	38	19	<i>1379 ± 73</i>
TinyViT-11M [16]	81.5	95.8	4000	11.0	224	300	2509	4	69	-	<i>1202 ± 191</i>
Mini-DeiT-S [14]	80.9	95.6	9400	11.0	224	300	1636	2	280	24	773 ± 16

Table 2: CViT performance on ImageNet-1K [7] compared to state-of-the-art lightweight and efficient ViT models. Throughput is tested on Nvidia V100-SXM2-32GB GPU, Intel Xeon Silver 4114 CPU @ 2.20GHz processor for CPU, Apple M4 Pro GPU for M4 Pro and AMD NPU for RyzenAI. Higher throughput indicates faster inference. Models are not traced before calculating throughput. FLOPs are calculated with fvcare [63]. Models are grouped by parameters and sorted in ascending order of FLOPs. Energy values in italic font indicate that a batch size of 1024 was used instead of 2048. We include Mini-DeiT-Ti and TinyViT-5M—even though they are slightly larger than the target parameter range—since their architectures pursue CViT-like objectives.

CViT’s accuracy/FLOP gains are modest, but its energy and memory savings are achieved over an already highly optimized SOTA baseline, so they’re non-trivial. By directly targeting edge bottlenecks, it is expected that CViT will deliver longer battery life or higher throughput, making it a strong deployment choice despite small accuracy deltas. These results render CViT an attractive solution for deployment in battery-powered and compute-constrained environments where a balance between efficiency and performance is critical.

4.3 Transfer Learning

We inspect the transferability of CViT by applying it to the COCO dataset [64] using an NVIDIA A100-40GB GPU for object detection and segmentation.

4.3.1 Object Detection

Table 3 presents the transfer learning results for object detection on COCO [64] dataset. Following the setup of EfficientViT [5], we set the learning rate to 1.5e-4, reduce the weight decay to 0.025, and use a batch size of 32. Under these conditions, CViT-L achieves an average precision (AP) of 29.5 with only 7M parameters, demonstrating its compactness and competitiveness. In comparison, EfficientViT achieves 3.2 AP higher but comes with a 1.8M parameter overhead and increased computational requirements. MicroViT-S3 performs better still, with 36.0 AP, but its 26.7M parameters result in substantial memory demands, making it less suitable for deployment in constrained environments.

CViT-L thus offers a compelling trade-off between accuracy, model size, and efficiency, making it an attractive option for edge applications. Notably, when trained on a 2× schedule, CViT-L improves to 31.4 AP, highlighting its capacity for further performance gains with extended training.

4.3.2 Instance Segmentation

Table 4 presents the transfer learning results on COCO [64] instance segmentation using Mask R-CNN [67] with a 1× schedule. We adopt the same training configuration used for object detection, except for a reduced learning rate

Model	RetinaNet 1x						Flops (M)	Params (M)
	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L		
CViT-L	29.5	48.4	30.5	15.7	31.1	40.5	249	7.0
EfficientViT-M4 [5]	32.7	52.2	34.1	17.6	35.3	46.0	299	8.8
MicroViT-S3 [41]	36.0	56.6	38.2	-	-	-	159	26.7
PVT-Tiny [65]	36.7	56.9	38.9	22.6	38.8	50.0	1900	13.2
EdgeViT-XXS [15]	38.7	59.0	41.0	22.4	42.0	51.6	600	13.1
CViT-L (2x)	31.4	50.7	32.6	16.6	33.5	44.0	249	7.0

Table 3: Transfer learning results on COCO [64] object detection using a RetinaNet [66] 1x schedule.

Model	Mask R-CNN 1x						Flops (M)	Params (M)
	AP ^b	AP ₅₀ ^b	AP ₇₅ ^b	AP ^m	AP ₅₀ ^m	AP ₇₅ ^m		
CViT-L	29.1	50.2	29.7	28.0	47.2	29.1	249	7.0
EfficientViT-M4 [5]	32.8	54.4	34.5	31.0	51.2	32.2	299	8.8
LightViT-T [68]	37.8	60.7	40.4	35.9	57.8	38.0	-	28
PVT-Tiny [65]	36.7	59.2	39.3	35.1	56.7	37.3	-	32.9
EdgeViT-XXS [15]	39.9	62.0	43.1	36.9	59.0	39.4	-	23.8

Table 4: Transfer learning results on COCO [64] instance segmentation using a Mask R-CNN 1x schedule.

of 1.25e-4 (down from 1.5e-4). While models like PVT-Tiny [65] and EdgeViT-XXS [15] incur additional parameter overhead when adapted for segmentation, our CViT-L maintains a fixed parameter count across tasks. This makes it a more compact and resource-efficient choice, while still delivering competitive performance. Additionally, given the performance gains observed with extended training in object detection, we expect further improvements in AP scores for CViT-L under a 2x schedule.

4.4 Latency on iPhone 15 Pro

We measure the per-image inference latency of our CViT models, along with several state-of-the-art baselines, on an Apple iPhone 15 Pro (See Table 5). All measurements are obtained using Apple’s Xcode Instruments profiler to ensure accurate timing under realistic mobile deployment conditions. The results reported below provide a fair comparison of mobile-side efficiency, highlighting how CViT balances accuracy with responsiveness on a modern smartphone.

We note that CViT consistently outperforms the backbone. Particularly, CViT-XL is 15% faster than EfficientViT-M5. While the improvements over the backbone may seem incremental for other scales (CViT-S, CViT-M and CViT-L), we emphasize that these are per image latencies. Small reductions at this level accumulate significantly, directly translating to longer battery life or faster inferences when models are deployed at scale or used for prolonged sessions.

We consider the comparison between CViT-S and EfficientViT-M0 to further illustrate the impact of the gains of our model. The latency gap between CViT-S and EfficientViT-M0 is a mere 0.03 ms per image. However, this translates to roughly 30 seconds of saved compute per million inferences. In the case of CViT-XL it is 130 seconds or 2 minutes and 10 seconds of saved compute. These gains translate to better energy efficiency which is critical for battery-powered edge-deployment.

CViT-based models also demonstrate favorable latency–accuracy trade-offs compared to other leading architectures. For instance, CViT-M matches the accuracy of MobileViTV2-0.5 while offering a faster inference time by 0.15 ms per image. CViT-XL attains accuracy on par with EfficientFormerV2-S0 but delivers a substantial 30% reduction in latency. Moreover, it surpasses SHViT-S2 by 0.3% in accuracy and reduces model size by 1.6M parameters, all while maintaining identical latency. Taken together, these results underscore that CViT not only achieves meaningful efficiency gains but also delivers improvements in accuracy and compactness. Such characteristics make CViT especially appealing for mobile and edge deployments, where even millisecond-level latency reductions compound into tangible performance and energy savings.

4.5 Accuracy Per Flop

[69] advocates for increased research focus on resource- and energy-efficient AI practices, such as using smaller datasets and designing computationally efficient models. This call is particularly relevant given the prevailing trend in model development that prioritizes accuracy and scale, often at the cost of significant energy consumption. For example,

Model	Top-1 (%)	Params (M)	Latency (ms/image)
CViT-S	62.0	1.9	0.39
EfficientViT-M0 [5]	63.2	2.3	0.42
EdgeNeXt-XXS [62]	71.2	1.3	0.51
MobileViTV2-0.5 [61]	70.2	1.4	0.60
CViT-M	69.9	3.5	0.45
EfficientViT-M2 [5]	70.8	4.2	0.48
EdgeViT-XXS [15]	74.4	4.1	0.80
EfficientFormerV2-S0 [17]	75.7	3.5	1.22
SHViT-S1 [60]	72.8	6.3	0.58
CViT-L	73.0	7.0	0.70
EfficientViT-M4 [5]	74.3	8.8	0.79
EdgeViT-XS [15]	77.2	6.7	1.25
SHViT-S2 [60]	75.2	11.4	0.86
CViT-XL	75.5	9.8	0.86
EfficientViT-M5 [5]	77.1	12.4	0.99

Table 5: Measured latency comparison of CViT and other efficient architectures on Apple iPhone 15 Pro. Latency is collected using XCode Instruments (Version 16.2) with CoreML.

[70] estimates the training cost of GPT-3 [71] to be approximately 50 MWh, which corresponds to the lifetime carbon emissions of five gasoline-powered cars. These concerns underscore the importance of evaluating model efficiency not solely in terms of predictive accuracy but also in relation to computational cost.

To address this, we propose a new metric aligned with the principles of “Green AI” [69], termed **Accuracy Per FLOP (APF)**, inspired by [72]. APF measures the accuracy gained per Mega-FLOP (MFLOP), offering a quantifiable indicator of a model’s computational efficiency. A higher APF indicates a model delivers more predictive performance per unit of compute, while a lower APF suggests inefficiency. We note from several studies that the relationship between accuracy and FLOPs is non-linear [73, 74, 75] where accuracy tends to saturate with increasing compute or FLOPs. Therefore, it is necessary to design a metric that does not assume a linear accuracy-FLOP relationship and thereby favor models that achieve superficial accuracy with very low compute.

To address this we first apply an accuracy threshold ($\geq 70\%$ Top-1 on ImageNet) following [76] to ensure that only models that achieve meaningful accuracy are included in comparative studies. In contrast to [76] where the energy term is anchored to the model with the highest energy, we aim to eliminate APF’s dependence on any model. Therefore, we apply log-scaling to the denominator—the log-scaled FLOP term grows more slowly in metric as raw FLOP count increases, capturing the relationship between accuracy and FLOPs observed in [73, 74, 75]. This results in a model-agnostic normalization that more faithfully reflects the non-linear relationship between accuracy and FLOPs. APF highlight models that achieve high accuracy with efficient compute without disproportionately penalizing larger models that necessarily operate in the high-compute, diminishing-return regime, as would be the case with a linear formulation. Formally APF is defined as

$$\text{APF} = \frac{\text{Top-1 Accuracy (\%)}}{\log_{10} \text{MFLOP}}. \quad (6)$$

Our CViT family consistently ranks among the most compute efficient models. We include CViT-M in this study as its 69.9% accuracy is $\approx 70\%$ given the $\pm 0.1\%$ variation from seeds and data shuffling, and it matches the performance of MobileViTV2-0.5 [61] on ImageNet-1K [7] while being 15% more efficient. CViT-L attains an APF of 30.5 with accuracy comparable to SHViT-S1 and to EfficientViT-M2 at a similar APF. CViT-XL and EfficientFormerV2-S0 achieve similar accuracy; however, CViT-XL yields 10% higher APF, indicating a stronger accuracy-compute trade-off. Across sizes, CViT delivers $\sim 2\%$ higher Top-1 than its EfficientViT backbone at comparable compute.

Model	Top-1 (%)	FLOPs (M)	APF
CoCa (SOTA) [6]	91.0	4540	24.9
CViT-M	69.9	173	31.2
EfficientViT-M2	70.8	201	30.7
MobileViTV2-0.5	70.2	480	26.2
EdgeNeXt-XXS	71.2	261	29.5
Mini-DeiT-Ti	73.0	2600	21.4
SHViT-S1	72.8	241	30.6
CViT-L	73.0	249	30.5
EfficientViT-M4	74.3	299	30.0
EdgeViT-XXS	74.4	600	26.8
SHViT-S2	75.2	366	29.3
CViT-XL	75.5	435	28.6
EfficientFormerV2-S0	75.7	800	26.1
EfficientViT-M5	77.1	522	28.4
SHViT-S3	77.4	601	27.9
EdgeViT-XS	77.2	1100	25.4

Table 6: Comparison of Top-1 accuracy, FLOPs, and Accuracy Per FLOP (APF) for all evaluated models where larger APF means greater efficiency.

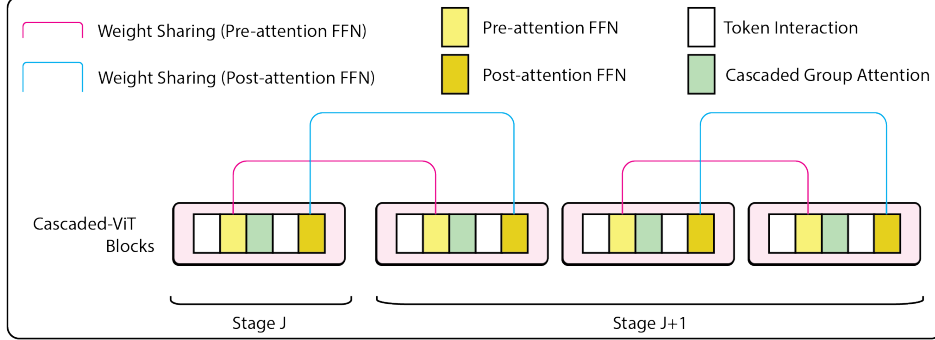


Figure 4: An overview of weight sharing. Specifically, the pre-attention FFN modules in successive Cascaded-ViT blocks share weights, and similarly, the post-attention FFN modules also share weights across blocks. Although Cascaded-ViT blocks are organized into different stages, the current implementation applies weight sharing across stage boundaries, i.e., between all successive blocks regardless of stage. When the total number of blocks is odd, the final block does not participate in weight sharing.

4.6 Ablation

We conduct an ablation study to investigate the impact of various design elements within the CCFFN module on the performance of our CViT architecture using the ImageNet-1K dataset [7]. Each experiment involves training CViT-M for 80 epochs to balance result maturity with computational efficiency. The sole exception is Experiment 1, where we use CViT-S and train for 300 epochs to ensure that performance differences are clearly distinguishable and representative.

We begin by evaluating the effectiveness of the CCFFN module as a compression mechanism by replacing it with a standard weight-sharing FFN and observing its effect on accuracy and parameter count. Subsequently, we analyze the

Model	No.	Ablation		Epochs	Top-1 (%)	Params (M)	FLOPs (M)	Energy/img (mJ)
		CCFFN	Wt. Shr.					
CViT-S	0	✓	✗	300	59.5	1.9	67	471 ± 29
	1	✗	✓	300	58.5	2.2	88	506 ± 10
CViT-M		No. Chunks	Exp. Ratio					
	2	2	2.5	80	62.83	3.5	173	571 ± 13
	3	2	4	80	63.80	4.2	201	591 ± 19
	4	4	2.5	80	60.80	2.9	150	555 ± 13
	5	4	4	80	61.54	3.2	164	561 ± 16
		Cascade	Project					
	6	✓	✓	80	58.7	3.9	192	581 ± 19
	7	✗	✗	80	62.60	3.5	173	578 ± 16

Table 7: Ablation of our Cascaded-ChunkFFN module and design choices for CViT-M. Ablation No. 1 is performed on CViT-S. Wt. Shr. means clustered weight sharing.

influence of two key hyperparameters: the number of chunks and the expansion ratio within the Chunk-FFN submodules. Finally, we assess the utility of cascading and projecting intermediate outputs within the CCFFN structure.

The results of these ablation experiments are summarized in Table 7, providing empirical insights into the design trade-offs and guiding principles for building efficient transformer-based models under resource constraints.

Effectiveness of CCFFN. Experiment 1 evaluates a clustered weight-sharing mechanism as an alternative to the proposed CCFFN module. Since full-scale weight sharing can severely constrain model flexibility and degrade performance [20], we adopt a more moderate strategy by grouping FFNs in pairs and sharing weights within each pair, as suggested in [20]. To enhance this setup, the expansion ratio of the final post-attention FFN layer is increased from $2\times$ to $4\times$. However, this configuration leads to a substantial increase in both FLOPs and parameter count, with energy consumption rising by 7.6%. In contrast, the CCFFN module achieves a 1% higher Top-1 accuracy while significantly reducing FLOPs, parameter count, and energy usage. For this experiment, CCFFN used a configuration of two chunks and a $2.5\times$ expansion ratio, selected based on results from subsequent experiments.

Optimal Number of Chunks and Expansion Ratio. Experiments 2 to 5 explore various combinations of chunk count and expansion ratios to identify the best trade-off between accuracy, model size, and energy efficiency. The configuration with two chunks and a $2.5\times$ expansion ratio achieves the most favorable balance and is adopted as the default across our model family. Increasing the expansion ratio to $4\times$ (Experiment 3) yields a marginal $\sim 1\%$ accuracy gain but comes at a significant cost in parameters, FLOPs, and energy. Conversely, increasing the number of chunks to four (Experiments 4 and 5) reduces computational cost and energy usage but results in a notable drop in classification accuracy. These results suggest that the two-chunk, $2.5\times$ configuration provides the best overall efficiency.

Impact of Cascading and Projection. Experiments 6 and 7 examine the benefits of cascading and projecting intermediate outputs in the CCFFN structure using the previously selected efficient configuration. When projection is applied, the model experiences a steep decline in accuracy, likely due to insufficient gradient flow in the projection branch. Removing cascading altogether, and instead processing FFNs in parallel, results in a 0.23% accuracy drop and increased energy consumption. These results confirm that the cascading mechanism contributes positively to both performance and energy efficiency.

5 Conclusion

In this paper, we introduced a novel Vision Transformer architecture, **CViT**, which enhances the efficiency of Feed Forward Networks (FFNs) through the proposed Cascaded ChunkFFN (CCFFN) modules. Our model achieves competitive Top-1 classification accuracy on the ImageNet-1K dataset [7] while offering significant reductions in parameter count, FLOPs, and energy consumption—making it well-suited for deployment in resource-constrained environments. Despite its strengths, CViT exhibits certain limitations. Specifically, the initialization of multiple small FFNs within CCFFN can introduce latency overhead on GPUs due to increased kernel launches. Additionally, the division of the input feature map into smaller chunks may lead to a slight reduction in model capacity, potentially impacting accuracy. As part of future work, we aim to improve the classification performance of CViT while preserving its compute and energy efficiency. This includes exploring more expressive chunk processing strategies and adaptive chunking mechanisms, as well as incorporating lightweight attention modules to enhance model capacity without significantly increasing overhead.

References

- [1] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.
- [2] Xiaoguang Tu, Zhi He, Yi Huang, Zhi-Hao Zhang, Ming Yang, and Jian Zhao. An overview of large ai models and their applications. *Visual Intelligence*, 2(1):1–22, 2024.
- [3] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [4] 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.
- [5] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14430, 2023.
- [6] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.
- [7] 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, 2009.
- [8] Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. Reversible column networks. *arXiv preprint arXiv:2212.11696*, 2022.
- [9] Eric Youn, Sanjana Prabhu, Siyuan Chen, et al. Compressing vision transformers for low-resource visual learning. *arXiv preprint arXiv:2309.02617*, 2023.
- [10] You Wu, Xucheng Wang, Xiangyang Yang, Mengyuan Liu, Dan Zeng, Hengzhou Ye, and Shuiwang Li. Learning occlusion-robust vision transformers for real-time uav tracking. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 17103–17113, June 2025.
- [11] Anish Bhattacharya, Nishanth Rao, Dhruv Parikh, Pratik Kunapuli, Yuwei Wu, Yuezhao Tao, Nikolai Matni, and Vijay Kumar. Vision transformers for end-to-end vision-based quadrotor obstacle avoidance. *arXiv preprint arXiv:2405.10391*, 2024.
- [12] Tristan Robitaille and Xilin Liu. Vision transformer accelerator asic for real-time, low-power sleep staging. *arXiv preprint arXiv:2502.16334*, 2025.
- [13] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 12934–12949. Curran Associates, Inc., 2022.
- [14] Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12145–12154, 2022.
- [15] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *European conference on computer vision*, pages 294–311. Springer, 2022.
- [16] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision (ECCV)*, 2022.
- [17] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems*, 35:12934–12949, 2022.
- [18] Telmo Pessoa Pires, António V Lopes, Yannick Assogba, and Hendra Setiawan. One wide feedforward is all you need. *arXiv preprint arXiv:2309.01826*, 2023.
- [19] Jianzong Wang, Ziqi Liang, Xulong Zhang, Ning Cheng, and Jing Xiao. Efficientasr: Speech recognition network compression via attention redundancy and chunk-level ffn optimization. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2024.
- [20] Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020.

- [21] Zehua Pei, Lancheng Zou, Hui-Ling Zhen, Xianzhi Yu, Wulong Liu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. CmoE: Converting mixture-of-experts from dense to accelerate llm inference. *arXiv preprint arXiv:2502.04416*, 2025.
- [22] Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. A survey on transformer compression. *arXiv preprint arXiv:2402.05964*, 2024.
- [23] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18547–18557, 2023.
- [24] Chuanyang Zheng, Kai Zhang, Zhi Yang, Wenming Tan, Jun Xiao, Ye Ren, Shiliang Pu, et al. Savit: Structure-aware vision transformer pruning via collaborative optimization. *Advances in Neural Information Processing Systems*, 35:9010–9023, 2022.
- [25] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *European conference on computer vision*, pages 620–640. Springer, 2022.
- [26] 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.
- [27] Mingjian Zhu, Yehui Tang, and Kai Han. Vision transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021.
- [28] 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.
- [29] Banafsheh Saber Latibari, Houman Homayoun, and Avesta Sasan. Optimizing vision transformers: Unveiling ‘focus and forget’ for enhanced computational efficiency. *IEEE Access*, 13:27908–27927, 2025.
- [30] Yuhong Song, Weiwen Jiang, Bingbing Li, Panjie Qi, Qingfeng Zhuge, Edwin Hsing-Mean Sha, Sakyasingha Dasgupta, Yiyu Shi, and Caiwen Ding. Dancing along battery: Enabling transformer with run-time reconfigurability on mobile devices. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1003–1008. IEEE, 2021.
- [31] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19(143-155):18, 1989.
- [32] Md Kowsher, Nusrat Jahan Prottasha, and Chun-Nam Yu. Does self-attention need separate weights in transformers? *arXiv preprint arXiv:2412.00359*, 2024.
- [33] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.
- [34] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.
- [35] Etienne Dupuis, David Novo, Ian O’Connor, and Alberto Bosio. Cnn weight sharing based on a fast accuracy estimation metric. *Microelectronics Reliability*, 122:114148, 2021.
- [36] Yiming Wang and Jinyu Li. Residualtransformer: Residual low-rank learning with weight-sharing for transformer layers. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11161–11165. IEEE, 2024.
- [37] Asifullah Khan, Zunaira Rauf, Anabia Sohail, Abdul Rehman Khan, Hifsa Asif, Aqsa Asif, and Umair Farooq. A survey of the vision transformers and their cnn-transformer based variants. *Artificial Intelligence Review*, 56(Suppl 3):2917–2970, 2023.
- [38] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31, 2021.
- [39] Benjamin 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. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12259–12269, 2021.
- [40] Shakti N Wadkar and Abhishek Chaurasia. Mobilevitv3: Mobile-friendly vision transformer with simple and effective fusion of local, global and input features. *arXiv preprint arXiv:2209.15159*, 2022.
- [41] Novendra Setyawan, Chi-Chia Sun, Mao-Hsiu Hsu, Wen-Kai Kuo, and Jun-Wei Hsieh. Microvit: A vision transformer with low complexity self attention for edge device. *arXiv preprint arXiv:2502.05800*, 2025.

- [42] Sia Gholami. Can pruning make large language models more efficient? In *Redefining Security With Cyber AI*, pages 1–14. IGI Global, 2024.
- [43] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [44] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [45] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.
- [46] 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.
- [47] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
- [48] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.
- [49] Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *2017 20th conference of the oriental chapter of the international coordinating committee on speech databases and speech I/O systems and assessment (O-COCOSDA)*, pages 1–5. IEEE, 2017.
- [50] Yi Liu, Pascale Fung, Yongsheng Yang, Christopher Cieri, Shudong Huang, and David Graff. Hkust/mts: A very large scale mandarin telephone speech corpus. In *International Symposium on Chinese Spoken Language Processing*, pages 724–735. Springer, 2006.
- [51] Soumyalatha Naveen and Manjunath R Kounte. Memory optimization at edge for distributed convolution neural network. *Transactions on Emerging Telecommunications Technologies*, 33(12):e4648, 2022.
- [52] Guanghe Cheng, Zhong Wan, Wenkang Ding, and Ruirui Sun. Memory allocation strategy in edge programmable logic controllers based on dynamic programming and fixed-size allocation. *Applied Sciences*, 13(18):10297, 2023.
- [53] Martin Maas, Ulysse Beaunon, Arun Chauhan, and Berkin Ilbeyi. Telamalloc: Efficient on-chip memory allocation for production machine learning accelerators. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 123–137, 2022.
- [54] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [55] 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.
- [56] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [57] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.
- [58] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.
- [59] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. *arXiv preprint arXiv:2205.10536*, 2022.
- [60] Seokju Yun and Youngmin Ro. Shvit: Single-head vision transformer with memory efficient macro design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5756–5767, June 2024.
- [61] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022.
- [62] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *European conference on computer vision*, pages 3–20. Springer, 2022.

- [63] FAIR. fvcore: Light-weight core library that provides the most common and essential functionality shared in various computer vision frameworks. <https://github.com/facebookresearch/fvcore>, 2019. [Online; accessed 2025-04-26].
- [64] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision—ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.
- [65] 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.
- [66] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.
- [67] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [68] Tao Huang, Lang Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Lightvit: Towards light-weight convolution-free vision transformers. *arXiv preprint arXiv:2207.05557*, 2022.
- [69] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.
- [70] Enrico Barbierato and Alice Gatti. Toward green ai: A methodological survey of the scientific literature. *IEEE Access*, 12:23989–24013, 2024.
- [71] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Nee-lakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [72] Andrea Esposito. “accuracy per flop”: A green ai metric for fair and efficient ai development. 2024.
- [73] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.
- [74] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.
- [75] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [76] Zeyu Yang, Karel Adamek, and Wesley Armour. Double-exponential increases in inference energy: The cost of the race for accuracy. *arXiv preprint arXiv:2412.09731*, 2024.