CaRe: Critical Parameter Rectification for Efficient Visual Modeling

Mingjia Li[1], Hongkun Xiong[1], Yuheng Shi[2], Hengxing Liu[1], and Xiaojie Guo[1] *

[1] School of Computer Software, Tianjin University, Tianjin, China
[2] Department of Computer Science, The University of Sydney, Sydney, Australia
{mingjiali,xionghongkun}@tju.edu.cn, yuheng.shi@sydney.edu.au,
{chrisliu.jz, xj.max.guo}@gmail.com

Abstract. Designing efficient vision backbones is increasingly challenging. An intriguing avenue is mergeable re-parameterization, which can reshape training dynamics by introducing auxiliary parameterizations without inference overhead. In particular, RepVGG-style structures can be viewed as adding a fixed-support correction on top of a dense base kernel after merging. In this paper, we propose **C**ritical P**a**rameter **Re**ctification (CaRe), which generalizes fixed corrections to learned selective rectification. CaRe factorizes each layer as $\mathbf{W} = \mathbf{W}_{\text{base}} + \mathbf{W}_{\text{aux}} \odot \phi(\mathbf{W}_{\text{gate}})$, where a static parameter-space gate induces sparse activation in the correction term, concentrating updates on a small subset of critical parameters during training. Since the gate depends only on parameters, CaRe is losslessly mergeable into a standard weight tensor, incurring zero inference overhead. Built on CaRe, the CaReNet family establishes a strong accuracy-throughput Pareto frontier on ImageNet-1K, with consistent gains across model scales and strong transfer to downstream dense prediction tasks. Code is available at `https://github.com/lime-j/care`.

Keywords: Vision Backbone · Efficient Network · Reparameterization

1 Introduction

The pursuit of efficient neural networks for deployment on resource-constrained devices continually pushes the boundaries of model design. Alongside architectural innovations, the underlying methods for parameterizing these networks are crucial as well. Structural re-parameterization has emerged as a particularly attractive approach [10, 11, 13, 14], offering increased training-time capacity that can be collapsed into an inference-time equivalent with no runtime overhead.

This paradigm enables better training dynamics that improve performance without impacting deployment efficiency. It can also be viewed as adding a correction kernel to a dense base weight. A representative work is RepVGG [14], which combines a 3×3 convolution with a 1×1 branch and an identity branch; after merging, the 1×1 and identity terms can be embedded into a 3×3 kernel

* Corresponding Author

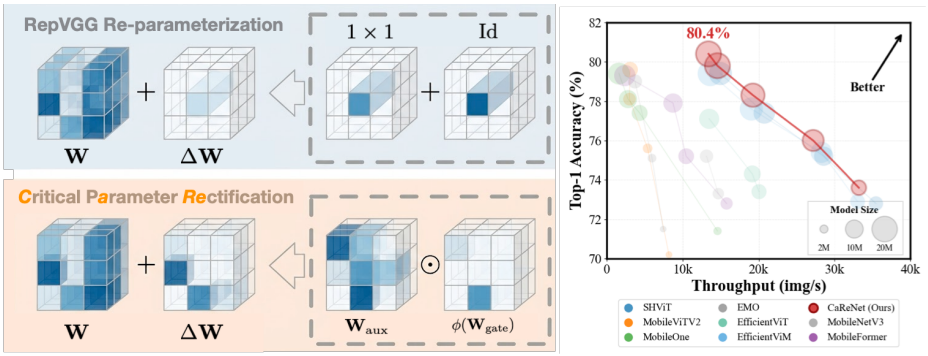

Fig. 1: An overview of our Critical Parameter Rectification. *Left.* RepVGG corresponds to a fixed-support sparse correction (center-only). CaRe learns a flexible sparse support via parameter-space gating and remains mergeable into a standard weight tensor. *Right.* We compare CaReNet with state-of-the-art efficient vision backbones on ImageNet-1K. Our CaReNet series establishes a new *Pareto frontier* (red curve), achieving higher accuracy at a comparable throughput level.

whose non-zero support is fixed at the center location. In other words, RepVGG implicitly adds a fixed-support sparse correction on top of a dense 3×3 base, as in Figure 1. This interpretation reveals a limitation: why should parameter corrections be confined to a predefined, static location? What if the model could learn *which* parameters are most critical to adjust?

This question motivates our work. Instead of structural re-parameterization that hard-codes the position of correction, we propose to generalize it to a *learned, selective rectification*. In this paper, we introduce Critical Parameter Rectification (CaRe), a mergeable parameterization that empowers a model to identify and amend its most impactful weights during training. CaRe factorizes each layer's weight tensor as a dense base plus a static, nonlinearly gated correction:

$$\mathbf{W} = \mathbf{W}_{\text{base}} + \mathbf{W}_{\text{aux}} \odot \phi(\mathbf{W}_{\text{gate}}). \tag{1}$$

Here, the parameter-space gate, $\phi(\mathbf{W}_{\text{gate}})$, acts as a learned on/off switch that induces sparse activation. This encourages the model to concentrate its updates on a small, dynamically identified subset of *critical parameters*, leaving the robust base largely intact. Crucially, because the gate depends only on static parameters and not the input features, CaRe is losslessly mergeable into a standard weight tensor before deployment, incurring absolutely zero inference overhead.

To demonstrate the practical impact of CaRe, we build CaReNet, a compact vision backbone co-designed at both macro and micro levels. At the macro level, CaReNet incorporates refinements over the previous SoTA, EfficientViM [27], to improve performance without sacrificing throughput. At the micro level, CaRe injects nonlinear, selectively activated capacity during training while remaining fully deployable after merging. Together, these designs improve the accuracy-throughput trade-off across ImageNet-1K and benefit diverse downstream tasks.

Extensive experiments validate the intended dense-plus-sparse behavior of CaRe and show that the activated critical parameters contribute to performance.

2 Related Work

2.1 Efficient Visual Architectures

Reducing computation and memory cost has long been central to on-device vision. Early efficient backbones rely on operator-level designs such as depthwise separable convolutions [6, 23] and channel shuffling [57], but their locality bias can limit long-range modeling [52]. Vision Transformers (ViTs) [15] introduce global self-attention, yet standard attention scales as $O(L^2)$, motivating windowed attention [34] and hardware-aware variants [32, 54]. Despite progress, attention often remains memory- and compute-intensive at high resolutions, and the practical efficiency of linear attention depends heavily on implementation. Recently, State Space Models (SSMs) [7, 16] offer a scalable alternative via selective scanning with $O(L)$ complexity. Vision adaptations such as Vision Mamba [58] and VMamba [33], along with follow-up works [26, 27, 38, 41, 51], further improve accuracy and efficiency through better 2D tokenization, scanning patterns, and hardware-oriented implementations. VSSD [40] involves the state-space duality in vision modeling, further improving the performance and processing speed. In a similar spirit, CaReNet adopts a lightweight linear-attention-style token mixer, but with a simpler design that achieves comparable performance.

2.2 Structural Re-parameterization

Structural re-parameterization decouples the training-time architecture from the inference-time structure. RepVGG [14], ACNet [11], and DBB [12] construct multi-branch topologies during training and fuse them into a single convolution for inference via linear superposition. RepMLPNet [10] extends the concept to MLP-based models. MobileOne [45] and RepViT [46] extend structural re-parameterization to mobile-friendly architectures. However, these methods are strictly confined to linear additive transformations to ensure mergeability. They create a fixed sparse addition into the base weight. In contrast, our CaRe induces sparsity in the addition within the optimization process, creating a flexible generalization of structural re-parameterization. By operating on static parameters, CaRe retains zero inference overhead while offering superior performance.

2.3 Gating Mechanisms and Dynamic Networks

Gating mechanisms dynamically modulate network activations based on input context. Feature-space gating methods, such as Squeeze-and-Excitation [25] and CBAM [49], recalibrate channel or spatial features. The Gated Linear Unit [8] and its variants SwiGLU [39] have become standard in modern LLMs. Dynamic convolutions, such as CondConv [50] and Dynamic Conv [5], generate input-dependent weights ($\mathbf{W} = \sum_i \pi_i(x)\mathbf{W}_i$). While powerful, all these methods operate

in a data-dependent manner, requiring element-wise operations or multiple weight banks at runtime. This inevitably increases memory access cost and inference latency to achieve a better expressivity. CaRe can be seen as a parameter-space gating technique, but the goals are different at its core. CaRe gates the parameter to create a non-symmetric landscape that aids optimization. CaRe does not introduce additional cost during inference and is compatible with the feature-space gating technique for further performance gain.

2.4 Weight Decomposition

Decomposing weight matrices has been a common practice in parameter-efficient fine-tuning to adapt large pre-trained models with minimal trainable parameters. Low-Rank Adaptation (LoRA) [24] is a representative approach, motivated by the hypothesis that fine-tuning updates lie in a low intrinsic dimension and parameterized as $W = W_0 + BA$. Building on this formulation, LoRA+ [17] improves optimization by using different learning rates for the A and B factors. Subsequent methods further extend the PEFT paradigm to increase flexibility, e.g., AdaLoRA [56] dynamically allocates rank across layers, GLoRA [2] unifies scaling and shifting in a more expressive form, and DoRA [31] decouples updates into magnitude and direction to better match full fine-tuning behavior. However, these methods are adaptation-centric: they assume a high-quality pre-trained weight W_0 and focus on learning an additive/auxiliary update during fine-tuning. As a result, weight decomposition is typically treated as an external adaptation module rather than a parameterization for joint training from scratch. Decomposition as a training-time mechanism for compact models remains relatively understudied.

3 CaRe: Critical Parameter Rectification

As mentioned before, we want to generalize Structural re-parameterization, which hard-codes a dense base with a fixed-support correction, to learn which parameters require correction. We aim to decompose the weight as $\mathbf{W}_{\text{base}} + \Delta\mathbf{W}$, where $\Delta\mathbf{W}$ acts as a sparse, selectively activated rectification tensor. To achieve this, we must force the network to concentrate its capacity on a small subset of critical parameters, leaving the robust base largely intact. However, a naive additive decomposition ($\mathbf{W}_{\text{base}} + \Delta\mathbf{W}$) is under-constrained. Without structural intervention, the optimization process will arbitrarily split learning capacity across both terms. This may cause the auxiliary component to degenerate into a redundant copy of the base. Therefore, the core design challenge is: *how can we introduce a structural inductive bias that strictly penalizes widespread non-zero entries in $\Delta\mathbf{W}$, naturally driving it toward sparsity during standard optimization?*

3.1 Design for Sparsity

It have been a well-established optimization principle that applying an ℓ_1 penalty encourages sparse solutions [1]. However, direct apply ℓ_1 regularization on the

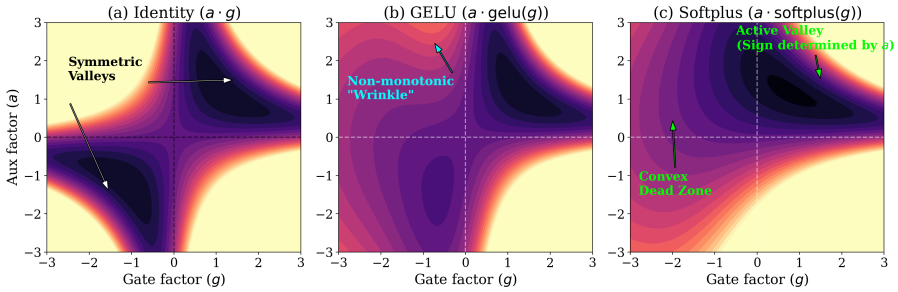

Fig. 2: Toy Optimization Landscapes. We plot the loss landscape for fitting a target $c = 1.5$ under default weight decay. (a) Identity ($\mathbf{a} \cdot \mathbf{g}$): Creates symmetric valleys and causes sign ambiguity. (b) GELU: Breaks symmetry but introduces a non-monotonic area (red arrow) that may trap gradients. (c) Softplus: Creates a monotonic half-pipe. The sign is controlled by $\mathbf{a}$, while $\mathbf{g}$ acts as a smooth and convex pruning gate.

weight makes the optimization non-differentiable at zero, which often makes standard gradient-based optimization complicated [21]. To induce sparsity without relying on such ℓ_1 regularizers, we draw inspiration from the sparsity by redundancy principle [59]. By over-parameterizing the correction term as a product of factors and applying standard ℓ_2 weight decay, we can seamlessly realize an ℓ_1-type sparsity bias. Consider a simplified scalar scenario where we design the correction as a linear factorization, say, $\Delta = a\,g$. With a standard weight decay penalty of $\frac{\lambda}{2}(a^2 + g^2)$, maintaining a non-zero correction incurs a predictable cost, as demonstrated in Lemma 1.

Lemma 1 (*Factorized L_2 induces L_1 on the effective correction*): For any target correction $\Delta \in \mathbb{R}$, optimizing $\min_{a,g\in\mathbb{R}}\{a^2 + g^2 : ag = \Delta\}$ reaches $2|\Delta|$.
Proof: By the AM-GM inequality, for arbitrary a and g, we have $a^2 + g^2 \geq 2|ag| = 2|\Delta|$. Equality is achieved by choosing $|a| = |g| = \sqrt{|\Delta|}$ with matching signs so that $ag = \Delta$. Since L_2 weight decay is element-wise, say,

$$\frac{\lambda}{2}(||\mathbf{W}_{\text{aux}}||_F^2 + ||\mathbf{W}_{\text{gate}}||_F^2) = \frac{\lambda}{2}\sum_{i,j}[(\mathbf{W}_{\text{aux}})_{ij}^2 + (\mathbf{W}_{\text{gate}})_{ij}^2],$$

applying the AM-GM inequality independently to each (i, j) yields $\sum_{i,j} 2|\Delta\mathbf{W}_{ij}| = 2||\Delta\mathbf{W}||_1$. Applied element-wise ($\Delta W_i = a_i g_i$), the minimum penalty resolves to $2\|\Delta\mathbf{W}\|_1$. This reveals that factorizing the correction assigns a linear cost to $|\Delta|$. This theoretical guarantee forms the foundation of our design. We thus penalizing dense corrections by factorization.

3.2 From Linear Factorization to Non-linear Gating

While Lemma 1 guarantees that a linear factorization $\Delta\mathbf{W} = \mathbf{a} \odot \mathbf{g}$ induces an ℓ_1-like penalty, it introduces sign ambiguity. Consider fitting a single scalar

weight to a target c. Under linear factorization, the loss landscape is $\mathcal{L} = (ag - c)^2 + \lambda(a^2 + g^2)$. As shown in Figure 2(a), this can create symmetric, hyperbolic valleys. The optimizer can reach the target via either $(a > 0, g > 0)$ or $(a < 0, g < 0)$. During training, different mini-batches may push the parameters toward opposite valleys, causing them to oscillate around the saddle point at the origin. Furthermore, the roles of $\mathbf{a}$ and $\mathbf{g}$ are entangled; say, both control the magnitude, and both fight over the sign.

To resolve this and stabilize the sparsity, we upgrade the linear factor $\mathbf{g}$ to a static non-linear gate. We define the Critical Parameter Rectification (CaRe) as:

$$\mathbf{W} = \mathbf{W}_{\text{base}} + \mathbf{W}_{\text{aux}} \odot \phi(\mathbf{W}_{\text{gate}}), \tag{2}$$

where $\odot$ denotes element-wise multiplication, $\mathbf{W}_{\text{aux}}$ acts as the magnitude carrier ($\mathbf{a}$), and $\mathbf{W}_{\text{gate}}$ acts as the sparsity controller ($\mathbf{g}$). $\phi(\cdot)$ is a fixed non-linear activation function. By using a strictly positive, monotonically increasing function; in the work, we use Softplus ($\phi(x) = \ln(1 + e^x)$), we can reshape the optimization landscape (Figure 2c). Because $\phi(g)$ is strictly positive, $\mathbf{a}$ assumes total control over the sign and magnitude of the correction. The symmetry is broken, and the optimizer is guided into a single, continuous active valley instead of two.

When a parameter is non-critical, $\mathbf{g}$ is pushed negative, driving $\phi(g) \to 0$. In this region, the data gradient vanishes, and the ℓ_2 weight decay gracefully pulls the parameters toward a stable off-state in a perfectly convex bowl. Moreover, unlike GELU [20] (Figure 2b), which contains a non-monotonic dip that can trap gradients, Softplus is strictly monotonic. If a pruned parameter suddenly becomes critical for a new task, the gradient $\phi'(g)$ provides a smooth path to re-enable the parameter.

Moreover, since this parameter-space gate depends on just static weights instead of dynamic input features, CaRe have zero inference overhead. Before deployment, we perform a one-time offline operation to collapse the components into $\mathbf{W} = \mathbf{W}_{\text{base}} + \mathbf{W}_{\text{aux}} \odot \phi(\mathbf{W}_{\text{gate}})$, operating exactly as a conventional layer.

4 Proof-of-Concept: CaReNet

To confirm the efficacy of CaRe, we proposed a proof-of-concept that builds upon EfficientViM [27], which combines convolutional locality with SSM-inspired global mixing. To keep our model simple, we simplify the SSD token mixer into a variant of channel-wise attention. We also changed the learnable multistage feature fusion with a fixed progressive strategy.

Overall architecture. CaReNet follows a simple multi-stage pyramid. Given an input image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, a convolutional stem downsamples it by a factor of 16 to produce the initial feature map $\mathbf{X}_0 \in \mathbb{R}^{D_0 \times \frac{H}{16} \times \frac{W}{16}}$. We then apply three stages, where stage i contains d_i repeated blocks and a downsampling module that halves spatial resolution and expands channels: $\mathbf{X}_{i+1} = \textsc{Down}(\mathbf{X}_i)$. This yields a feature hierarchy with increasing semantic abstraction. For classification, we attach lightweight heads to multiple stages and fuse their predictions; the macro architecture can be found in Figure 3.

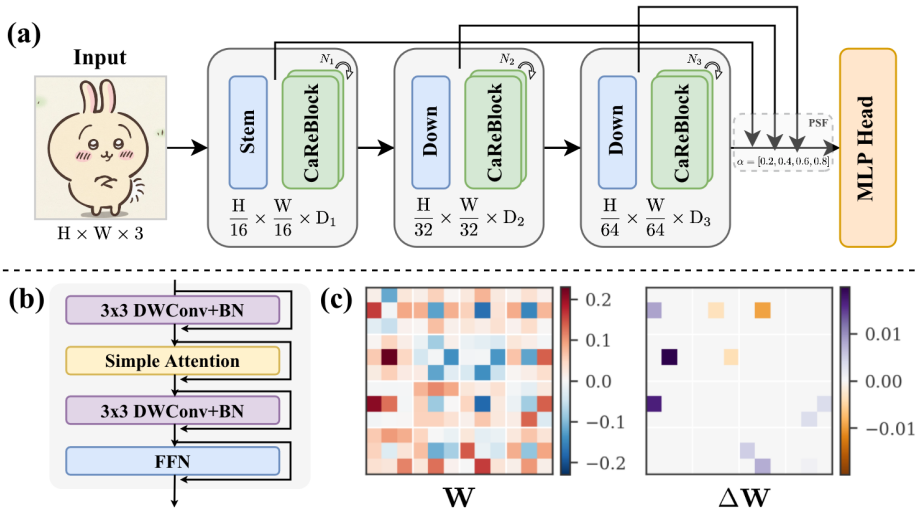

Fig. 3: **(a)** Overall architecture of our CaReNet. **(b)** Detailed design of the CaReBlock. Note that all Depthwise Convolution (DWConv) and Feedforward Network (FFN) layers within the block are enhanced with our CaRe re-parameterization. **(c)** Visualized depthwise kernels (16 channels) from a Simple Attention layer in CaReNet-M1, demonstrating that the CaRe mechanism successfully induces a sparse set of $\Delta\mathbf{W}$.

Simple Attention. Inspired by MambaOut [53], we distill the token mixing mechanism in EfficientViM into two core components: a *global-state* linear attention operator for efficient cross-channel information mixing, and a parallel gating mechanism at the output. Unlike standard softmax attention, which scales quadratically $\mathcal{O}(L^2)$ with the sequence length L, our streamlined design constructs a compact, fixed-size global context matrix. Remarkably, replacing the HSM-SSD module in EfficientViM-M1 with our Simple Attention yields matched empirical performance. This observation suggests that heavily parameterized state-space models may not be strictly necessary in this regime; instead, a lightweight global linear aggregation coupled with local convolutions is sufficient to capture the required representations. Formally, given an input feature map $\mathbf{X} \in \mathbb{R}^{B\times C\times L}$, we first generate the key ($\mathbf{K}$) and query ($\mathbf{Q}$) embeddings. To inject a necessary local spatial inductive bias prior to global mixing, we apply a 3×3 depthwise convolution followed by a SiLU activation. This process can be formulated as:

$$[\mathbf{K}, \mathbf{Q}] = \sigma\big(\mathrm{DWConv}(\mathbf{W}_{kq}\mathbf{X})\big) \in \mathbb{R}^{B\times 2S\times L}, \tag{3}$$

where $[\cdot,\cdot]$ denotes the split operation along the channel dimension, S is the compact state dimension, $\mathbf{W}_{kq}$ represents the initial linear projection weights, and $\sigma(\cdot)$ denotes the SiLU function. Concurrently, a parallel linear projection generates the value tensor $\mathbf{V}$ and a spatial gating tensor $\mathbf{Z} \in \mathbb{R}^{B\times D\times L}$, where D is the expanded inner dimension. We then perform the core $\mathcal{O}(L)$ linear attention by first aggregating the sequence into a global context state $\mathbf{G}$, and subsequently

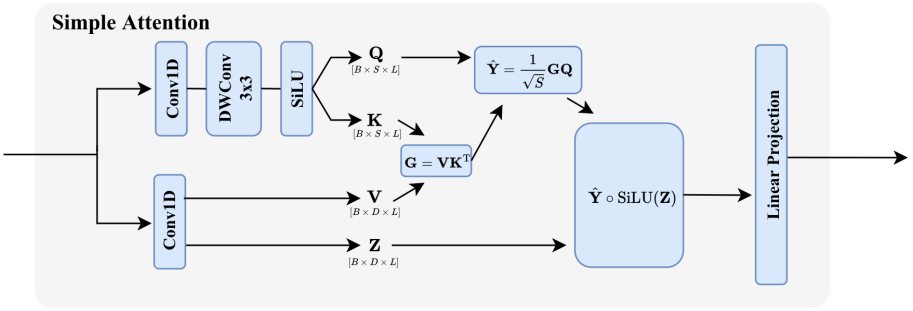

Fig. 4: Architecture of the Simple Attention module. Our Simple Attention scales linearly with the sequence length L while avoiding expensive softmax operations. It processes the input through two parallel branches: one extracts queries and keys equipped with a depthwise convolution for local spatial bias, while the other generates values and a gating signal. A compact global state is constructed to mix information efficiently, followed by a GLU-style gating mechanism at the end of the block.

distributing it back to each spatial position:

$$\mathbf{G} = \mathbf{V}\mathbf{K}^\top \in \mathbb{R}^{B\times D\times S}, \qquad \hat{\mathbf{Y}} = \frac{1}{\sqrt{S}}\mathbf{G}\mathbf{Q} \in \mathbb{R}^{B\times D\times L}. \tag{4}$$

An illustration of the Simple Attention can be found in Figure 4. Unlike standard softmax attention, Eq. (4) avoids expensive spatial normalization, allowing $\mathbf{G}$ to act as a consolidated memory state that summarizes the entire sequence. Finally, to recover the representational capacity typically lost when removing the softmax non-linearity, we employ a GLU-style channel modulation using the parallel gate $\mathbf{Z}$, followed by a linear output projection:

$$\mathbf{Y} = \mathbf{W}_{out}\big(\hat{\mathbf{Y}} \odot \sigma(\mathbf{Z})\big) \in \mathbb{R}^{B\times C\times L}, \tag{5}$$

where $\odot$ denotes element-wise multiplication.

Progressive Stage Fusion. Unlike EfficientViM, which employs a learnable weighted sum followed by a softmax for multi-stage fusion, we observe that such dynamic weighting prematurely converges to a sub-optimal selection, over-relying on the final stage. We propose Progressive Stage Fusion (PSF) with a fixed linear progression ([0.2,0.4,0.6,0.8]). This ensures that spatial cues from earlier stages are consistently integrated with deep semantic states.

5 Experimental Validation

5.1 Implementation Details

We train all variants from scratch on the ImageNet-1K dataset [9] for 300 epochs, except for the 450 epoch M4 model. To ensure a strict and fair comparison, our

training recipe closely follows that of EfficientViM [27]. Specifically, we employ the AdamW optimizer with a peak learning rate of 2.5×10^{-3} and a cosine annealing schedule that gradually decays to 2.0×10^{-5}. For training stability, we incorporate a 20-epoch linear warmup, a 10-epoch cooldown period, and gradient clipping with a maximum norm of 0.02. Additionally, an exponential moving average (EMA) with a decay rate of 0.9995 is applied to the model weights to obtain the final evaluation checkpoints. The model is train on a TPU v4-8 with a batch size of 512 per device, resulting in a batch size of 2048 in total. For downstream object detection and instance segmentation, we initialize from the 300-epoch ImageNet-pretrained checkpoint and follow the EfficientViM hyperparameters, fine-tuning for 12 epochs, *i.e.*, the standard $1\times$ schedule using the AdamW optimizer with a learning rate of 1.0×10^{-4}.

Following EMO, SHViT and EfficientViM, our CaReNet family is also designed with a three-stage hierarchical macro-architecture. In each stage, we use multiple CaRe-parameterized blocks. To be specific, the CaReNet architecture consists of four variants, each defined by its specific number of blocks, channel dimensions, and hidden state sizes. Across all four models, the hidden state size remains constant at [64, 32, 16]. The first three variants (CaReNet-M1, M2, and M3) all share a standard block configuration of [2, 2, 2], differing only in their channel capacities: M1 uses [128, 192, 320], M2 increases to [128, 256, 512], and M3 further expands to [224, 320, 512]. The final variant, CaReNet-M4, utilizes the channel dimensions as M3 but features a deeper block configuration of [3, 4, 2].

5.2 ImageNet Classification

As we depicted in Table 1, by replacing conventional static weights with the proposed CaRe mechanism, CaReNet achieves a clear performance lead over the latest SOTA, EfficientViM [27]. Specifically, our CaReNet-M1 variant achieves an impressive 73.6% Top-1 accuracy, yielding a +0.7% margin over EfficientViM-M1 (72.9%) while maintaining a higher throughput of 33,203 im/s. We observe similar gains in the mid-range category, where CaReNet-M3 provides a +0.7% accuracy improvement (78.3% vs. 77.6%) over its EfficientViM counterpart at a comparable inference speed. Compared to hardware-optimized models like SHViT-S1 [54], CaReNet-M1 provides a 0.8% precision boost (73.6% vs. 72.8%). The scalability of our approach is further evidenced by CaReNet-M4, which reaches 79.8% accuracy, surpassing the heavily optimized MobileOne-S4 [45] (79.4%) while operating nearly *9.0*× faster (14,576 vs. 1,624 im/s). Finally, at 256×256 resolution, CaReNet-M4-256 achieves a peak accuracy of 80.4%, outperforming EfficientViM-M4 by a significant 1.0% margin. Following EfficientViM, we also report a model trained under 450 epoches. These consistent improvements across various scales and resolutions confirm that the structural decomposition in CaRe serves as an efficient generative mechanism that effectively bridges the gap between model capacity and execution speed through weight-space rectification.

Table 1: Comparison of efficient networks on ImageNet-1K [9] classification. The throughput is benchmarked with a NVIDIA A100-PCIE-40GB GPU.

Method	Venue	Input Size	Throughput (im/s) ↑	Top-1 (%) ↑	Params (M)	FLOPs (M)
MobileViTV2 0.5 [36]	Arxiv'22	256^2	8,199	70.2	1.4	466
MobileOne-S0 [45]	CVPR'23	224^2	14,584	71.4	2.1	275
EMO-1M [55]	ICCV'23	224^2	7,435	71.5	1.3	261
MobileFormer-96M [4]	CVPR'22	224^2	15,780	72.8	4.6	96
SHViT-S1 [54]	CVPR'24	224^2	35,450	72.8	6.3	241
EfficientViM-M1 [27]	CVPR'25	224^2	33,033	72.9	6.7	239
MobileNetV3-L 0.75 [22]	ICCV'19	224^2	14,687	73.3	4.0	155
EfficientViT-M3 [32]	CVPR'23	224^2	20,043	73.4	6.9	263
CaReNet-M1 (Ours)	-	224^2	**33,203**	**73.6**	6.7	264
EfficientFormerV2-S0 [28]	NeurIPS'22	224^2	970	73.7	3.5	407
EfficientViT-M4 [32]	CVPR'23	224^2	19,164	74.3	8.8	299
EdgeViT-XXS [37]	ECCV'22	224^2	5,661	74.4	4.1	556
EMO-2M [55]	ICCV'23	224^2	5,966	75.1	2.3	439
MobileNetV3-L 1.0 [22]	ICCV'19	224^2	13,139	75.2	5.4	217
MobileFormer-151M [4]	CVPR'22	224^2	10,463	75.2	7.6	151
SHViT-S2 [54]	CVPR'24	224^2	28,559	75.2	11.4	366
EfficientViM-M2 [27]	CVPR'25	224^2	28,249	75.4	13.9	355
MobileViTV2 0.75 [36]	Arxiv'22	256^2	5,393	75.6	2.9	1030
CaReNet-M2 (Ours)	-	224^2	**27,180**	**76.0**	13.9	391
EfficientMod-XXS [35]	ICLR'24	224^2	6,364	76.0	4.7	583
ConvNeXtV2-A [48]	CVPR'23	224^2	6,155	76.2	3.7	552
EfficientViT-M5 [32]	CVPR'23	224^2	13,488	77.1	12.4	522
MobileOne-S2 [45]	CVPR'23	224^2	4,336	77.4	7.8	1299
SHViT-S3 [54]	CVPR'24	224^2	20,744	77.4	14.2	601
EdgeViT-XS [37]	ECCV'22	224^2	4,600	77.5	6.7	1136
EfficientViM-M3 [27]	CVPR'25	224^2	19,038	77.6	16.6	656
MobileFormer-294M [4]	CVPR'22	224^2	8,732	77.9	11.4	294
CaReNet-M3 (Ours)	-	224^2	**19,249**	**78.3**	16.9	727
ConvNeXtV2-F [48]	CVPR'23	224^2	5,334	78.0	5.2	785
MobileViTV2 1.0 [36]	Arxiv'22	256^2	3,001	78.1	4.9	1844
MobileOne-S3 [45]	CVPR'23	224^2	2,816	78.1	10.1	1896
EfficientMod-XS [35]	ICLR'24	224^2	5,292	78.3	6.6	778
EMO-6M [55]	ICCV'23	224^2	3,725	79.0	6.1	961
MobileFormer-508M [4]	CVPR'22	224^2	2,396	79.3	14.0	508
MobileOne-S4 [45]	CVPR'23	224^2	1,624	79.4	14.8	2978
SHViT-S4 [54]	CVPR'24	256^2	14,975	79.4	16.5	986
EfficientViM-M4 [27]	CVPR'25	256^2	13,625	79.4	19.6	1111
MobileViTV2 1.25 [36]	Arxiv'22	256^2	3,086	79.6	7.5	2857
CaReNet-M4 (Ours)	-	224^2	**14,576**	**79.8**	19.6	964
CaReNet-M4-256 (Ours)	-	256^2	**13,404**	**80.4**	19.6	1223
CaReNet-M4-256-450ep (Ours)	-	256^2	**13,404**	**80.6**	19.6	1223

5.3 Downstream Tasks: Detection & Segmentation

To evaluate the transferability and generalization of CaReNet, we integrate it as a backbone into the Mask R-CNN [18] and RetinaNet [29] frameworks for instance segmentation and object detection, respectively. All experiments are conducted on the COCO-2017 [30] dataset using the MMDetection library [3], following the standard 1× training schedule (12 epochs).

Table 2: Downstream evaluations. Instance segmentation with Mask R-CNN head and object detection with RetinaNet on COCO-2017 [30].

Instance Segmentation with Mask R-CNN [18]						
Method	$\mathbf{AP}^{\mathrm{b}}$	$\mathbf{AP}^{\mathrm{b}}_{50}$	$\mathbf{AP}^{\mathrm{b}}_{75}$	$\mathbf{AP}^{\mathrm{m}}$	$\mathbf{AP}^{\mathrm{m}}_{50}$	$\mathbf{AP}^{\mathrm{m}}_{75}$
EfficientNet-B0 [42]	31.9	51.0	34.5	29.4	47.9	31.2
PoolFormer-S1 [52]	37.3	59.0	40.1	34.6	55.8	36.9
FastViT-SA12 [44]	38.9	60.5	42.2	35.9	57.6	**38.1**
SHViT-S4 [54]	39.0	**61.2**	41.9	35.9	**57.9**	37.9
EfficientViM-M4 [27]	**39.3**	60.2	42.5	35.8	57.1	37.4
CaReNet-M4	**39.3**	**61.2**	**42.6**	**36.0**	**57.9**	37.8
Object Detection with RetinaNet [29]						
Method	**AP**	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$	$\mathbf{AP}_{\mathrm{s}}$	$\mathbf{AP}_{\mathrm{m}}$	$\mathbf{AP}_{\mathrm{l}}$
PVTV2-B0 [47]	37.2	57.2	39.5	**23.1**	40.4	49.7
MobileFormer-508M [4]	38.0	58.3	40.3	22.9	41.2	49.7
EdgeViT-XXS [37]	38.7	59.0	41.0	22.4	42.0	51.6
SHViT-S4 [54]	38.8	**59.8**	41.1	22.0	42.4	52.7
EfficientViM-M4 [27]	38.8	59.6	41.1	22.1	42.4	52.8
CaReNet-M4	**39.2**	59.7	**42.0**	21.9	**42.7**	**53.0**

Object Detection. Table 2 summarizes the performance of various backbones within the RetinaNet framework. CaReNet-M4 achieves **39.2** AP, outperforming both the state-of-the-art EfficientViM-M4 [27] and the hardware-efficient SHViT-S4 [54] by a margin of 0.4 AP. Notably, our backbone exhibits superior performance in detecting large objects, reaching **53.0** AP_L. This surpasses EfficientViM-M4 (52.8 AP_L), underscoring the effectiveness of our architectural design in capturing complex, high-level semantic features. Furthermore, CaReNet provides significant boosts of +0.5 AP and +2.0 AP over other competitive models like EdgeViT-XXS [37] and PVTV2-B0 [47], while maintaining high computational efficiency. These gains demonstrate that the CaRe mechanism successfully captures the rich spatial hierarchies and long-range dependencies essential for precise object localization.

Instance Segmentation. Within the Mask R-CNN [18] framework, CaReNet establishes superior performance benchmarks for instance segmentation. As reported in Table 2, our model achieves **39.3** AP^b and **36.0** AP^m. Compared to the competitive EfficientViM-M4, CaReNet matches its bounding box accuracy while delivering a notable improvement in mask prediction (+0.2 AP^m). Furthermore, CaReNet consistently outperforms other high-performance models, including SHViT-S4 [54], FastViT-SA12 [44], and PoolFormer-S1 [52], across nearly all key metrics such as AP^b_{50} and AP^b_{75}. These results validate that our CaRe generates more robust and geometrically precise representations for dense prediction tasks.

Table 3: Composition formula. We evaluate different ways to combine the components. Additive combination with multiplicative gating achieves the best performance.

Formula	**Top-1 (%)**	**Δ**
$\mathbf{W} = \mathbf{P}_1$ (Standard Conv)	72.7	–
Single Modification		
$\mathbf{W} = \mathbf{P}_1 \odot \phi(\mathbf{P}_2)$	69.5	-3.2
$\mathbf{W} = \mathbf{P}_1 + \mathbf{P}_3$	73.1	+0.4
Re-parameterization style		
$\mathbf{W} = \mathbf{P}_{3\times3} + \mathbf{P}_{1\times1} + \mathbf{P}_{id}$	73.2	+0.5
Gated Combinations		
$\mathbf{W} = (\mathbf{P}_1 \odot \phi(\mathbf{P}_2)) + (\mathbf{P}_3 \odot \phi(\mathbf{P}_4))$	70.1	-2.6
$\mathbf{W} = \mathbf{P}_1 \odot \phi(\mathbf{P}_2) + \mathbf{P}_3$, remove $\mathbf{P}_1 \odot \phi(\mathbf{P}_2)$ after training	60.1	-12.6
$\mathbf{W} = \mathbf{P}_1 \odot \phi(\mathbf{P}_2) + \mathbf{P}_3$ (**Ours**)	**73.6**	**+0.9**

6 Analysis and Discussion

In this section, we conduct ablation studies with ImageNet-1K classification task to validate the core design choices of our proposed architecture. Furthermore, we demonstrate that the CaRe mechanism can be integrated it into mainstream vision backbones and improve their performance. Unless otherwise specified, all ablation are under identical settings to make sure that the comparison is fair.

Composition Formula. We first investigate the formulation of our weight composition strategy. As shown in Table 3, we evaluate different methods of combining the decomposed components against a standard convolution baseline. Relying purely on a single parameter modification or standard linear re-parameterization yields sub-optimal results. Notably, removing the base weight entirely degrades performance, demonstrating the crucial role of static base weights in maintaining representational stability during training. The proposed additive combination with multiplicative gating ($\mathbf{W} = \mathbf{P}_1 \odot \phi(\mathbf{P}_2) + \mathbf{P}_3$) achieves the best performance, yielding a significant +0.9% Top-1 accuracy improvement over the standard convolution baseline. Furthermore, to verify whether the learned sparse residuals are genuinely critical, we evaluate the model by discarding the auxiliary component after training. This removal causes a severe performance collapse to 60.1% Top-1 accuracy, confirming that the selectively activated parameters encapsulate indispensable representations rather than redundant features.

Activation Function for Gating. The choice of the non-linear activation function ϕ is vital to the efficacy of the gating mechanism, as it reshapes the optimization landscape. As detailed in Table 6, removing ϕ (i.e., using an identity mapping) causes the mechanism to degenerate into a trivial linear re-parameterization, resulting in a sub-optimal baseline accuracy of 72.9%. While bounded functions (Sigmoid, Tanh) and standard rectified gating (ReLU, ELU) provide modest gains, smooth and strictly monotonic activations perform significantly better.

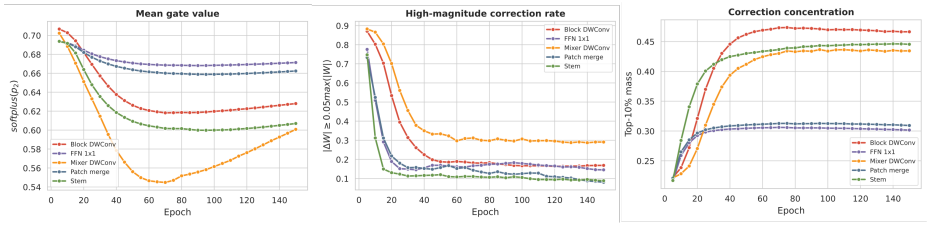

Fig. 5: Gate evolving dynamics during training. We track three statistics of the gated correction $\Delta\mathbf{W} = \mathbf{W}_{\text{aux}} \odot \phi(\mathbf{W}_{\text{gate}})$ across representative layer types. *Left:* the mean gate value $\phi(\mathbf{W}_{\text{gate}})$ stays strictly positive, confirming smooth soft modulation rather than binary pruning. *Middle:* the fraction of high-magnitude entries ($|\Delta\mathbf{W}| \geq 0.05 \max(|\mathbf{W}|)$) drops sharply, indicating that the correction is driven toward sparsity. *Right:* the share of total correction mass held by the top-10% largest entries rises, showing that the surviving corrections concentrate on a small set of critical parameters.

Specifically, Softplus provides superior representational flexibility and a more stable gradient path for weight modulation, achieving the highest Top-1 accuracy of 73.6%, a clear +0.7% improvement over the identity baseline.

Multi-stage Fusion Weights. Next, we compare different weighting configurations α for aggregating stage-wise representations. The results are summarized in Table 5. Relying solely on the final stage or applying a uniform average across all stages fails to fully exploit the hierarchical nature of the feature maps. We observe that while deeper stages encode richer semantic information, earlier stages still provide indispensable fine-grained structural context. Consequently, a *Progressive* weighting strategy ($\alpha = [0.2, 0.4, 0.6, 0.8]$), which assigns gradually increasing importance to deeper stages, achieves the optimal balance. This progression yields a peak Top-1 accuracy of 73.6%, outperforming the baseline by +0.5%.

Generality Across Architectures. To demonstrate that Critical Parameter Rectification (CaRe) is a universal, plug-and-play operator rather than an architecture-specific trick, we extend its application to a diverse set of representative vision paradigms. Table 4 summarizes the results of integrating CaRe into a standard CNN (ResNet-18 [19]), Transformer (ViT-Tiny [43] and MetaFormer (PoolFormerV2-S12 [52]) variants. For ResNet, we directly parameterize the standard 3×3 convolutional weights using CaRe. Beyond convolutions, CaRe readily adapts to the dense linear layers fundamental to Vision Transformers. For ViT and PoolFormer, we apply the CaRe decomposition to the linear projection matrices ($\mathbf{W} \in \mathbb{R}^{d_{out} \times d_{in}}$) within the Multi-Head Self-Attention (QKV projections) and Feed-Forward Network (FFN) blocks. The accuracy improvements across these different architectures confirm that flexibly activating critical parameters provides a widely applicable representational bias. Most importantly, since CaRe operates entirely on static weights during training, all performance gains are achieved with *zero increase in inference-time parameters or FLOPs.*

Table 4: Extension to other backbone architectures. We apply our CaRe to other arch under the DeiT training recipe. "Params" refers to the inference-time cost, which remains identical to the baseline due to our lossless merging mechanism.

Model	Params	Top-1 (%)	Δ
ResNet-18 [19]	11.7M	69.1	–
+ **CaRe**	11.7M	**69.3**	**+0.2**
ViT-Tiny [43]	5.7M	72.2	–
+ **CaRe**	5.7M	**72.4**	**+0.2**
PoolFormerV2-S12 [52]	11.9M	77.7	–
+ **CaRe**	11.9M	**78.1**	**+0.4**

Table 5: Multi-stage fusion weights. We compare different weighting configurations α for aggregating stage-wise hidden states on ImageNet-1K.

Strategy	Configuration α	Top-1 (%)	Δ
Final Stage Only	$[0, 0, 0, 1.0]$	73.1	–
Uniform Average	$[0.25, 0.25, 0.25, 0.25]$	72.9	-0.2
Simple Linear	$[0.1, 0.2, 0.4, 0.8]$	73.2	+0.1
Sparse Early	$[0.05, 0.1, 0.4, 1.0]$	73.1	+0.0
Steep Linear	$[0.1, 0.3, 0.7, 1.0]$	73.1	+0.0
Progressive	**[0.2, 0.4, 0.6, 0.8]**	**73.6**	**+0.5**

Gate Evolving Dynamics. To understand how the correction term goes over training, we track three statistics of the gated rectification across layer types (Stem, Patch merge, Block/Mixer DWConv, and FFN 1×1), as we demonstrated in Figure 5. *(i) Mean gate value.* The averaged gate output $\phi(\mathbf{W}_{\text{gate}})$ decreases during early training and then stabilizes, yet it remains continuous and strictly positive, instead of a binary $\{0, 1\}$ mask. This confirms that CaRe acts as a smooth, soft modulation rather than hard pruning. Some layers (e.g., Mixer DWConv) further exhibit a non-monotonic trajectory, first descending and then partially recovering, which indicates that parameters deactivated early can be smoothly re-enabled later. *(ii) High-magnitude correction rate.* Tracking the effective correction $|\Delta\mathbf{W}|$ rather than the gate alone, the fraction of entries with $|\Delta\mathbf{W}| \geq 0.05 \max(|\mathbf{W}|)$ drops sharply within the first $\sim$40 epochs and settles at a low plateau. This shows that optimization actively drives the correction tensor toward sparsity, leaving the dense base largely intact while restricting updates to a limited set of entries. *(iii) Correction concentration.* The share of total correction mass carried by the top-10% largest entries rises steadily and saturates at a high level, revealing that the surviving corrections are not merely sparse but become increasingly concentrated on a small subset of *critical parameters*.

Table 6: ϕ for gating. We evaluate activation choices within our CaRe mechanism.

Activation ϕ	**Output Range**	**Top-1 (%)**	**Δ**
None (Identity)	$(-\infty, +\infty)$	72.9	–
Linear and Rectified Gating			
ReLU	$[0, +\infty)$	73.1	+0.2
ELU	$(-1, +\infty)$	73.2	+0.3
Bounded Gating			
Sigmoid	$(0, 1)$	73.2	+0.3
Tanh	$(-1, 1)$	73.3	+0.4
Smooth Unbounded Gating			
SiLU (Swish)	$(\approx -0.28, +\infty)$	73.1	+0.2
GELU	$(\approx -0.17, +\infty)$	73.5	+0.6
Softplus	$(0, +\infty)$	**73.6**	**+0.7**

7 Conclusion

In this study, we introduce Critical Parameter Rectification (CaRe), a reparameterization technique that generalze and overcame the limitations of traditional, fixed-support structural adjustments. By factorizing weight tensors into a dense base and a selectively activated sparse correction, CaRe allows models to dynamically identify and optimize critical subset of parameters during training. Because this gating mechanism operates just within the parameter space, it collapses losslessly before deployment, incurring absolutely zero inference overhead. Extensive evaluations demonstrate the advantages of this approach. Based on CaRe, we built proof-of-concept architecture namely CaReNet. It establishes a new accuracy-throughput Pareto frontier on ImageNet-1K and exhibits strong transferability to downstream dense prediction tasks. Furthermore, we show that CaRe acts as a universal, plug-and-play operator. It can improves performance of standard CNNs and Vision Transformers without adding a single FLOP during inference. We believe that CaRe offers a powerful new tool for deploying highly accurate models on resource-constrained hardware and modern accelerators.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 62372251 and U2574209. Mingjia Li would like to express his gratitude to TPU Research Cloud (TRC), for their generous donation of computational resources. The authors would like to thank Hainuo Wang and Qiming Hu for the insightful discussions and feedback.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, 1 edn. (2007)
2. Chavan, A., Liu, Z., Gupta, D., Xing, E., Shen, Z.: One-for-all: Generalized lora for parameter-efficient fine-tuning. arXiv preprint arXiv:2306.07967 (2023)
3. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al.: Mmdetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
4. Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., Liu, Z.: Mobile-former: Bridging mobilenet and transformer. In: CVPR (2022)
5. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11030–11039 (2020)
6. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR (2017)
7. Dao, T., Gu, A.: Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In: International Conference on Machine Learning (ICML) (2024)
8. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 933–941. PMLR (2017)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
10. Ding, X., Chen, H., Zhang, X., Han, J., Ding, G.: Repmlpnet: Hierarchical vision mlp with re-parameterized locality. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 578–587 (2022)
11. Ding, X., Guo, Y., Ding, G., Han, J.: Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1911–1920 (2019)
12. Ding, X., Zhang, X., Han, J., Ding, G.: Diverse branch block: Building a convolution as an inception-like unit. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10886–10895 (2021)
13. Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11963–11975 (2022)
14. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: RepVGG: Making VGG-style convnets great again. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13733–13742 (2021)
15. Dosovitskiy, A.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021)
16. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv:2312.00752 (2023)
17. Hayou, S., Ghosh, N., Yu, B.: Lora+: Efficient low rank adaptation of large models. In: ICML (2024)
18. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)

20. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
21. Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., Peste, A.: Sparsity in deep learning: pruning and growth for efficient inference and training in neural networks. J. Mach. Learn. Res. **22**(1) (Jan 2021)
22. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: ICCV (2019)
23. Howard, A.G.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861 (2017)
24. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. In: ICLR (2022)
25. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7132–7141 (2018)
26. Huang, T., Pei, X., You, S., Wang, F., Qian, C., Xu, C.: Localmamba: Visual state space model with windowed selective scan. arXiv:2403.09338 (2024)
27. Lee, S., Choi, J., Kim, H.J.: Efficientvim: Efficient vision mamba with hidden state mixer based state space duality. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2025)
28. Li, Y., Hu, J., Wen, Y., Evangelidis, G., Salahi, K., Wang, Y., Tulyakov, S., Ren, J.: Rethinking vision transformers for mobilenet size and speed. In: ICCV (2023)
29. Lin, T.Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: 2017 IEEE International Conference on Computer Vision (ICCV) (2017)
30. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
31. Liu, S.Y., Wang, C.Y., Yin, H., Molchanov, P., Wang, Y.C.F., Cheng, K.T., Chen, M.H.: Dora: Weight-decomposed low-rank adaptation. In: ICML (2024)
32. Liu, X., Peng, H., Zheng, N., Yang, Y., Hu, H., Yuan, Y.: Efficientvit: Memory efficient vision transformer with cascaded group attention. In: CVPR (2023)
33. Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., Liu, Y.: Vmamba: Visual state space model. NeurIPS (2024)
34. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
35. Ma, X., Dai, X., Yang, J., Xiao, B., Chen, Y., Fu, Y., Yuan, L.: Efficient modulation for vision networks. ICLR (2024)
36. Mehta, S., Rastegari, M.: Separable self-attention for mobile vision transformers. arXiv:2206.02680 (2022)
37. Pan, J., Bulat, A., Tan, F., Zhu, X., Dudziak, L., Li, H., Tzimiropoulos, G., Martinez, B.: Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In: ECCV (2022)
38. Shaker, A., Wasim, S.T., Khan, S., Gall, J., Khan, F.S.: Groupmamba: Parameter-efficient and accurate group visual state space model. arXiv:2407.13772 (2024)
39. Shazeer, N.: GLU variants improve transformer. arXiv preprint arXiv:2002.05202 (2020), `https://arxiv.org/abs/2002.05202`
40. Shi, Y., Dong, M., Li, M., Xu, C.: Vssd: Vision mamba with non-casual state space duality. arXiv:2407.18559 (2024)
41. Shi, Y., Dong, M., Xu, C.: Multi-scale vmamba: Hierarchy in hierarchy visual state space model. NeurIPS (2024)

42. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019)
43. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
44. Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: Fastvit: A fast hybrid vision transformer using structural reparameterization. In: ICCV (2023)
45. Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: Mobileone: An improved one millisecond mobile backbone. In: CVPR (2023)
46. Wang, A., Chen, H., Lin, Z., Han, J., Ding, G.: Repvit: Revisiting mobile cnn from vit perspective. In: CVPR (2024)
47. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvtv2: Improved baselines with pyramid vision transformer (2022)
48. Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I.S., Xie, S.: Convnext v2: Co-designing and scaling convnets with masked autoencoders. In: CVPR (2023)
49. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: CBAM: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
50. Yang, B., Bender, G., Le, Q.V., Ngiam, J.: CondConv: Conditionally parameterized convolutions for efficient inference. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
51. Yang, C., Chen, Z., Espinosa, M., Ericsson, L., Wang, Z., Liu, J., Crowley, E.J.: Plainmamba: Improving non-hierarchical mamba in visual recognition. BMVC (2024)
52. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: CVPR (2022)
53. Yu, W., Wang, X.: Mambaout: Do we really need mamba for vision? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2025)
54. Yun, S., Ro, Y.: Shvit: Single-head vision transformer with memory efficient macro design. In: CVPR (2024)
55. Zhang, J., Li, X., Li, J., Liu, L., Xue, Z., Zhang, B., Jiang, Z., Huang, T., Wang, Y., Wang, C.: Rethinking mobile block for efficient attention-based models. In: ICCV (2023)
56. Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., Zuo, T.: Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In: ICLR (2023)
57. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: CVPR (2018)
58. Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X.: Vision mamba: Efficient visual representation learning with bidirectional state space model. ICML (2024)
59. Ziyin, L., Wang, Z.: spred: Solving L_1 penalty with SGD. In: Proceedings of the 40th International Conference on Machine Learning. pp. 43407–43422 (2023)