EfficientViM: Efficient Vision Mamba with Hidden State Mixer based State Space Duality

Supplementary Material

Configuration	Base	Distillation	FT
Epochs	300/450	300	30 + 30
Batch size	2048		1024
Weight decay	0.05		1e-8
Warmup Epochs	20		0
Cooldown Epochs	10		0
Learning rate	2e-3		1e-3
Min Learning rate	2e-5		1e-5
Optimizer (Momentum)	Adamw (0.9, 0.999)		
Gradient Clipping	0.02		
Learning rate scheduler	Cosine		
Rand Augment	rand-m9-mstd0.5-inc1		
Mixup	0.8		
Cutmix	1.0		
Mixup switch prob	0.5		
Random erasing prob	0.25		
Label smoothing	0.1		
EMA decay rate	0.9995		
Teacher model	None	RegNetY-160	None

Table A. **Settings for training EfficientViM.** FT: finetuning with higher resolution images (Section B).

A. Implementation Details

We use the ImageNet-1K [3] to validate the effectiveness of EfficientViM on the image classification task. For training EfficientViM, we follow the training recipes of previous works [21, 24, 28]. Specifically, all models are trained from scratch with a batch size of 2,048 for 300 epochs using AdamW optimizer [14] with a warmup of 20 epochs and a cooldown of 10 epochs. Following [1, 7, 10], we also report the results after training 450 epochs. During training, we adopt a cosine annealing [15] scheme with the initial learning rate of 2×10^{-3} decreasing to 2×10^{-5}. The weight decay of 0.05 and gradient clipping with a threshold of 0.02 are used. Also, MESA [4] and EMA with the decay rate of 0.9995 is adopted following [5, 21]. For data augmentation, we follow DeiT [23] using Mixup [30] & CutMix [29] with a Label smoothing [22], RandAugment [2], and Random Erasing [32]. We report the throughput and latency with the batch size of 256 on Nvidia RTX 3090 GPU.

Additionally, we finetune the model with the batch size of 1024, using cosine annealing with the initial learning rate of 1×10^{-3}, for 30 epochs at a resolution of 384^2, followed by an additional 30 epochs at 512^2. For a fair comparison, we employ pre-trained models trained for 300 epochs. Also, to report the throughput of the models with extremely

Method	Size	**Thr.**	**Thr$_{\text{rel}}$**	**Top-1**	Params	FLOPs
SHViT-S4 [28]	384^2	3,685	$\times$0.99	81.0	16.5M	2225M
EfficientViM-M4	384^2	3,724	$\times$1.00	80.9	21.3M	2379M
SHViT-S4 [28]	512^2	2,122	$\times$0.86	82.0	16.5M	3973M
EfficientViM-M4	512^2	2,452	$\times$1.00	81.9	21.3M	4154M

Table B. **Classification results on ImageNet-1K [3] after fine-tuning with higher resolutions.** **Thr$_{\text{rel}}$** is relative throughput compared to EfficientViM-M4.

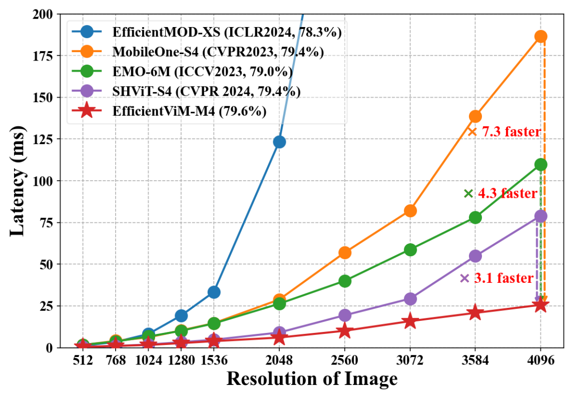


Figure A. **Latency comparison of recent efficient networks for an extremely high-resolution image.**

high-resolution images in Figure A, we start with a batch size of 256 and halve it once the memory exceeds the GPU limit as the resolution increases. Regarding training with distillation, all settings are the same as in Table 3. of the main paper except for the guidance from the teacher model of RegNetY-160 [20] following DeiT [23].

B. EfficientViM with high-resolution images.

Following [28], we also explore the applicability of EfficientViM on high-resolution images after fine-tuning 30 epochs at a resolution of 384^2, followed by an additional 30 epochs at 512^2. For a fair comparison, we use the EfficientViM-M4 pre-trained for 300 epochs. In 384^2 size, EfficientViM demonstrates competitive performance as presented in Table B. Interestingly, when the resolution increases, the throughput gap between EfficientViM and SHViT [28] gets larger, resulting in more than 15% speedup compared to SHViT in 512^2 while achieving comparable accuracy. We further investigate the scalability of our method in extremely high-resolution images beyond 512^2, by comparing the latency of the models while scaling the resolution from 512^2 to 4096^2. As depicted in Figure A, we observe an advantage of EfficientViM over the

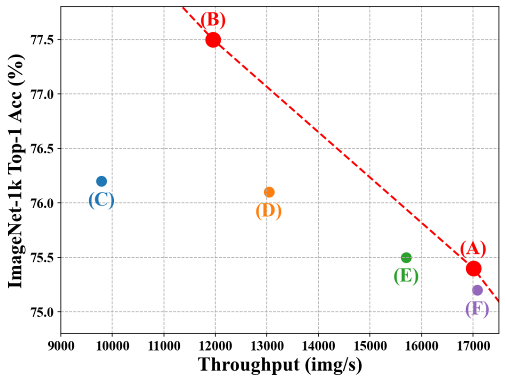
Method	Thr.	Top-1	Params	FLOPs
(A) **EfficientViM-M2** (Base)	17,005	75.4	13.9M	355M
(B) **EfficientViM-M3**	11,952	77.5	16.6M	656M
a. Token Mixers (Base: HSM-SSD)				
(C) (→) NC-SSD [21]	9,786	76.2	13.0M	382M
(D) (→) Self-Attention [27]	13,038	76.1	13.6M	416M
b. Head designs (Base: Single-head (SH) with $\mathbf{A} \in \mathbb{R}^{L\times N}$*)*				
(E) (→) Multi-head	15,703	75.5	13.9M	352M
(F) (→) SH w. $\mathbf{a} \in \mathbb{R}^{L}$	17,081	75.2	13.9M	352M
c. Multi-stage fusion (Base: $\mathbf{h}^{(s)}$*)*				
(G) (→) Fusion with $\mathbf{x}^{(s)}$	17,041	75.3	13.4M	355M
(H) (→) None	17,317	75.1	13.0M	354M
d. # states (N) of each stage (Base: $[49, 25, 9]$*)*				
(I) (→)$[9, 25, 49]$	16,476	75.4	14.0M	407M
(J) (→)$[25, 25, 25]$	16,991	75.2	13.9M	373M
e. Normalization (Base: Partial LN)				
(L) (→) Full BN	17,432	*NaN*	13.9M	355M

Table C. **Ablation studies on EfficientViM.** All ablations are conducted with EfficientViM-M2. See Figure B for a visualization comparing the ablated models with the Pareto front of EfficientViM.

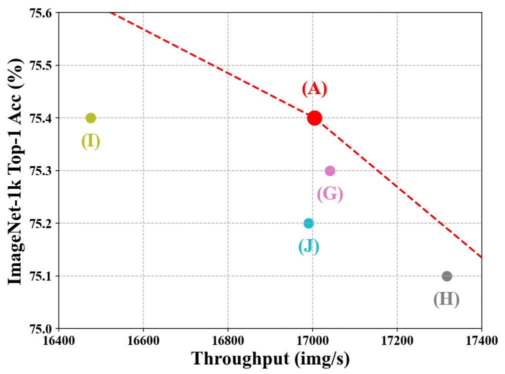
recent state-of-the-art method on extremely high-resolution images. EfficientViM shows about 3×, 4×, and 7× faster speed compared to SHViT, EMO [31], and MobileOne [26], respectively. This notable result highlights the scalability of EfficientViM for high-resolution images based on linear cost of HSM-SSD.

C. Ablation Studies

Here, we show the effectiveness of HSM-SSD by ablating the proposed components. The results are summarized in Table C, and Figure B. First, we compare HSM-SSD with other global token mixers, by replacing them with other methods including (C) NC-SSD [21] and (D) self-attention (SA) [27]. Considering that EfficientViM-M3 with HSM-SSD shows 77.5% with a throughput of 11,952 (im/s), NC-SSD and SA show a poorer speed-accuracy trade-off than HSM-SSD. Regarding head choices, we observe that our single-head design brings significant speed-up (17005 (im/s)) over (E) multi-head (15,703 (im/s)) without performance degradation. Additionally, defining the importance score per state as $\mathbf{A} \in \mathbb{R}^{L\times N}$ to mimic multi-head leads to the +0.2% gain with a minor increase in latency, compared to using the original score (F) $\mathbf{a} \in \mathbb{R}^{L}$. Note that all ablates models (C-F) are placed under the Pareto front of EfficientViM (Figure Ba), which proves the efficacy of HSM-SSD and our single-head design.
Also, multi-stage fusion with hidden states (75.4%) surpasses the accuracy of (G) the fusion with the output feature maps $\mathbf{x}^{(s)}$ (75.3%) and (H) the EfficientViM without fusion (75.1%) under similar throughput. For the number of states N, we observe that an increasing schedule with respect to the stages is more effective than (I) a decreasing or (J) constant schedule. See (G)-(J) in Table C and Figure Bb for the ablation studies on multi-stage fusion and the number of states. Additionally, for normalization, using (L) batch normalization (BN) across all operations is simple and fast, but this approach leads to numerical instability. Therefore, we apply layer normalization (LN) only before HSM-SSD, and BN for the rest.



(a) **Ablations on the token mixer and head design (C-F).**



(b) **Ablations on Multi-stage fusion and # states (G-J).**

Figure B. **Ablation studies on EfficientViM.** Refer to Table C for the corresponding models. Red line indicates the Pareto frontier of EfficientViM.

Method	Token Mixer	Thr.	Top-1	Params	FLOPs
VSSD-M [21]	NC-SSD	1459	82.5	14M	2.3G
VSSD-T [21]	NC-SSD	947	84.1	24M	4.5G
VSSD-T	→ **HSM-SSD**	1660	82.7	24M	3.7G

Table D. **Comparison of HSM-SSD with NC-SSD.**

D. Comparison of HSM-SSD with NC-SSD

To show the advantage of the proposed HSM-SSD over NC-SSD, we replace NC-SSD of VSSD-T [21] with HSM-SSD

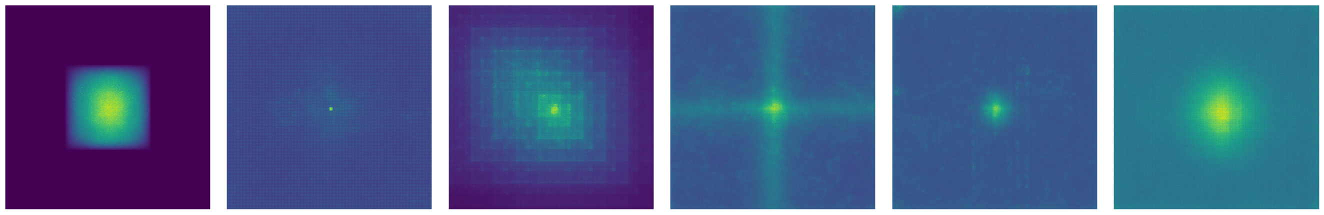


Figure C. **Comparison of Effective Receptive Fields (ERF) [16]**

and train the model following the original training configuration provided in the official repository. In Table D, after replacing the token mixer with HSM-SSD, VSSD-T with HSM-SSD demonstrates a significant increase in the throughput (1.8×). Notably, compared to scaling down the models to smaller sizes (*e.g.*, VSSD-M), replacing the token mixer with HSM-SSD provides better speed-accuracy trade-offs (14% faster, +0.2% accuracy), highlighting the advantage of HSM-SSD over NC-SSD. We also provide the qualitative comparison of HSM-SSD with NC-SSD in the following section.

E. Effective Receptive Field of HSM-SSD

In this section, we qualitatively compare the HSM-SSD with previous token mixers including convolutions in ResNet50 [6], attentions in vision Transformers such as DeiT-S [23] and Swin-T [13], and SSM and SSD variants in vision Mambas like VMamba-T [12] and VSSD-T [21]. We here analyze the Effective Receptive Fields (ERF) [16] of each model, which quantifies the region of the input that contributes to the output. In Figure C, we visualize the ERF with respect to the central pixel of the output feature maps. Among various models, EfficientViM-M4 with HSM-SSD shows a global receptive field rather than focusing on a specific region. For instance, ResNet50 shows a relatively small ERF due to its intrinsic locality of convolution. The attention mechanism in DeiT-S predominantly focuses on the central pixel itself, and the shifted window attention in Swin-T limits the global receptive field. Further, since SSM is conducted after flattening the image patch both vertically and horizontally in VMamba-T, it generates an unnatural cross pattern in ERF. VSSD-T shows a relatively better global receptive field, yet it still largely depends on the close region. On the other hand, EfficientViM-M4 generates a global effective receptive field (ERF) similar to that of VSSD-T but extracts more information from all regions, enabling it to capture the global dependencies better.

F. CPU & Mobile Latency

To understand the applicability of EfficientViM in a resource-constrained environment, we here provide the latencies of vision backbones in GPU, CPU, and mobile devices (Table E). The latencies are measured with a batch size of 256 on an NVIDIA RTX 3090 GPU, 16 on an AMD EPYC 7742 CPU, and 1 on an iPhone 16 (iOS 18.1). For mobile latencies, we use CoreML [19] library. EfficientViM-M4 achieves the highest accuracy of 79.6% with the lowest latency on GPUs and competitive latency on CPU and iPhone 16. Although EfficientViM-M4 shows slightly higher latency than a few of the prior works in CPU and mobile, EfficientViM-M4 shows a significantly lower latency as the resolution increases (Figure D). In the resolution of 2048^2, EfficientViM-M4 achieves at least 58% and 20% reductions in latency compared to previous works in iPhone 16 and CPU, respectively. To summarize, EfficientViM serves as a general solution suitable for both GPU and edge devices. Furthermore, EfficientViM is an effective backbone for real-world applications where high-resolution images are given, such as in image generation, object detection, and instance segmentation.

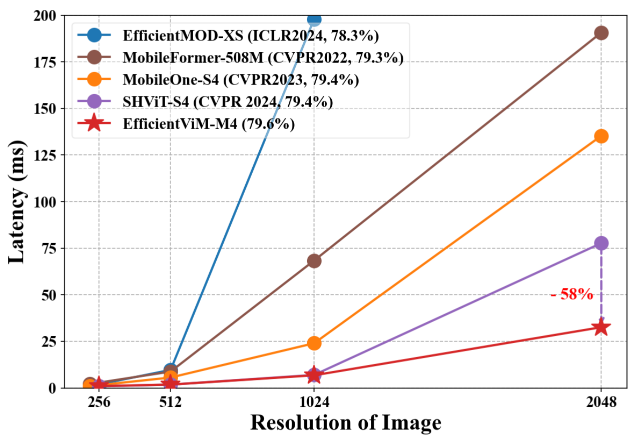
Method	Latency (ms)			**Top-1**
	GPU	CPU	Mobile	
MobileViTV2 1.0 [18]	0.34	138.8	1.1	78.1
EfficientMod-XS [17]	0.19	33.1	0.9	78.3 (79.4)
MobileFormer-508M [1]	0.22	29.7	2.1	79.3
FastViT-T12 [25]	0.37	81.3	1.8	79.1 (80.3)
MobileOne-S4 [26]	0.33	79.9	1.0	79.4
SHViT-S4 [28]	0.12	27.4	0.9	79.4 (80.2)
EfficientViM-M4	0.12	32.1	1.0	79.6 (80.7)

Table E. **Latency comparison of EfficientViM-M4 with prior works.** The number in parentheses indicates the performance with distillation.

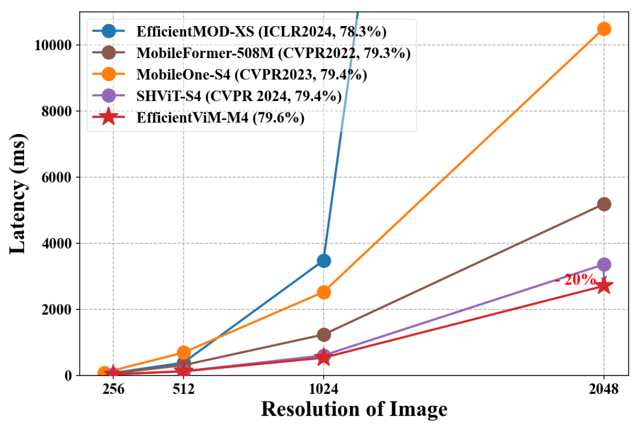
G. Proof for Proposition 1

Proposition 1. *Let* $N = L$, $\mathbf{a}\mathbb{1}_L^\top \odot \mathbf{B} = \mathbb{I}_L$, *and* $\mathbf{C} \in \mathbb{R}^{L\times L}$ *be diagonal. Then, HSM-SSD*$(\mathbf{x}, \mathbf{a}, \mathbf{B}, \mathbf{C})$ *is equivalent to NC-SSD*$(\mathbf{x}, \mathbf{a}, \mathbf{B}, \mathbf{C})$ *including gating and output projection, as* $\mathbf{x}_{out} = f(\mathbf{y}) = \mathbf{C}f(\mathbf{h})$.

Proof. It is sufficient to show that $\mathbf{C}f(\mathbf{h})$ of HSM-SSD is equivalent to $f(\mathbf{y}) = (\mathbf{Ch} \odot \sigma(\mathbf{x}_{\text{in}}\mathbf{W}_{\mathbf{z}}))\mathbf{W}_{\text{out}}$ in order to prove the proposition. Here, based on the assumption, the



(a) **Mobile latency comparison.**



(b) **CPU latency comparison.**

Figure D. **Mobile and CPU latency comparison.**

following holds:

$$\begin{aligned}
&\mathbf{C}f(\mathbf{h}) \\
&\quad = \mathbf{C}\left((\mathbf{h}\odot\sigma(\mathbf{h}_{\text{in}}\mathbf{W}_{\mathbf{z}}))\mathbf{W}_{\text{out}}\right) \\
&\quad = \mathbf{C}\left((\mathbf{h}\odot\sigma(((\mathbf{a}\mathbb{1}_L^\top\odot\mathbf{B})^\top\mathbf{x}_{\text{in}})\mathbf{W}_{\mathbf{z}}))\mathbf{W}_{\text{out}}\right) \\
&\quad = \mathbf{C}(\mathbf{h}\odot\sigma(\mathbf{x}_{\text{in}}\mathbf{W}_{\mathbf{z}}))\mathbf{W}_{\text{out}} \\
&\quad = (\mathbf{C}\mathbf{h}\odot\sigma(\mathbf{x}_{\text{in}}\mathbf{W}_{\mathbf{z}}))\mathbf{W}_{\text{out}} = f(\mathbf{y})
\end{aligned}$$

□

Remarks. We assume that $\mathbf{C}$ is a diagonal matrix but $\mathbf{C}$ is dependent on $\mathbf{x}_{\text{in}}$ since $\mathbf{C} = \mathbf{x}_{\text{in}}\mathbf{W}_{\mathbf{C}}$. Unfortunately, there does not exist a weight matrix $\mathbf{W}_{\mathbf{C}}$ that makes $\mathbf{C}$ diagonal for arbitrary inputs $\mathbf{x}_{\mathbf{in}}$. We provide this proposition to understand the relationship between the HSM-SSD and NC-SSD operations. This implies that in specific conditions with particular data, the two methods yield the same result. However, one approach does not generalize the other.

H. Discussion of multi-stage hidden stage fusion

In this section, we briefly discuss how multi-stage hidden state fusion (MSF) provides a performance boost, although the earlier layers generally provide less accurate logits. Note that our MSF leverages the logits across the layer as *deep supervision* and *multi-scale representation* to improve the performance. MSF aligns with the concept of 'deep supervision' in pioneering works, such as DSN [9], and U-Net++ [33]. Specifically, during training, MSF can be interpreted as auxiliary classification tasks, encouraging even the earlier layers to learn more discriminative features. Further, the earlier layers generally capture fine-grained patterns, while later layers extract high-level semantics, i.e., DenseNet [8] and FPN [11]. By combining these complementary representations with the learnable coefficients, we take advantage of the ensemble effect from 'multi-scale representations'. In fact, it is well-known that an ensemble of the models often outperforms each model, highlighting the benefits of HSM. As a results, MSF brings substantial improvements on EfficientViM.

References

[1] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *CVPR*, 2022. 1, 3

[2] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 1

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[4] Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. *NeurIPS*, 2022. 1

[5] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. *arXiv:2405.16605*, 2024. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[7] 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 *ICCV*, 2019. 1

[8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 4

[9] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015. 4

[10] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *ICCV*, 2023. 1

[11] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 4

[12] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *NeurIPS*, 2024. 3

[13] 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 *ICCV*, 2021. 3

[14] I Loshchilov. Decoupled weight decay regularization. *ICLR*, 2019. 1

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016. 1

[16] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *NeurIPS*, 2016. 3

[17] Xu Ma, Xiyang Dai, Jianwei Yang, Bin Xiao, Yinpeng Chen, Yun Fu, and Lu Yuan. Efficient modulation for vision networks. *ICLR*, 2024. 3

[18] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv:2206.02680*, 2022. 3

[19] Core ML. https://developer.apple.com/documentation/coreml, 2017. 3

[20] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 1

[21] Yuheng Shi, Minjing Dong, Mingjia Li, and Chang Xu. Vssd: Vision mamba with non-casual state space duality. *arXiv:2407.18559*, 2024. 1, 2, 3

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1

[23] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 3

[24] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *ECCV*, 2022. 1

[25] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *ICCV*, 2023. 3

[26] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Mobileone: An improved one millisecond mobile backbone. In *CVPR*, 2023. 2, 3

[27] A Vaswani. Attention is all you need. *NeurIPS*, 2017. 2

[28] Seokju Yun and Youngmin Ro. Shvit: Single-head vision transformer with memory efficient macro design. In *CVPR*, 2024. 1, 3

[29] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 1

[30] Hongyi Zhang. mixup: Beyond empirical risk minimization. *ICLR*, 2018. 1

[31] Jiangning Zhang, Xiangtai Li, Jian Li, Liang Liu, Zhucun Xue, Boshen Zhang, Zhengkai Jiang, Tianxin Huang, Yabiao Wang, and Chengjie Wang. Rethinking mobile block for efficient attention-based models. In *ICCV*, 2023. 2

[32] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. 1

[33] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *MICCAI Workshop*, 2018. 4