Supplementary Material - FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization

Pavan Kumar Anasosalu Vasu[†] James Gabriel Jeff Zhu Oncel Tuzel Anurag Ranjan[†]

Apple

Ablation	Description	Top-1 Acc.	Mobile Latency (ms)
-	Baseline	79.8	1.4
Normalization	BatchNorm → LayerNorm	79.7	1.7
Activation	GELU → ReLU	79.4	1.3
	GELU → SiLU	79.7	1.4
Hybrid Stages	[RepMix, RepMix, RepMix, SelfAttn.]	80.6	1.6
	[RepMix, RepMix, SelfAttn., SelfAttn.]	81.2	3.7
	[RepMix, SelfAttn., SelfAttn., SelfAttn.]	81.5	5.0
	[SelfAttn., SelfAttn., SelfAttn., SelfAttn.]	82.0	11.7

Table 1: Ablation for FastViT-S12 on ImageNet-1K. All models are trained and benchmarked using the same settings described in main paper.

A. Ablations

A.1. Architectural Choices

The primary motivation behind the design choices made for FastViT is efficient mobile deployment. The cost of self-attention blocks is very high, especially when there are many tokens. In early stages (when the number of tokens are high), self attention can be replaced with efficient alternatives with small accuracy degradation but significantly lower latency. In Table 4 of main paper, we analyzed this for last two stages. In Table 1, we present the full analysis for S12 architecture.

A.2. Large Kernel Convolutions

In Table 2, we ablate over various kernel sizes in FFN and patch embedding layers and report their Top-1 accuracy on ImageNet-1k along with mobile latency on iPhone 12 Pro. We observe that performance of the model stagnates beyond the kernel size of 7×7, while the overall FLOPs, latency and parameter count increases. Hence, we use 7×7 kernel size in our network architecture.

Kernel Size	Params (M)	FLOPs (G)	Mobile Latency(ms)	Top-1 (%)
3×3	8.7	1.7	1.3	78.9
5×5	8.7	1.8	1.4	79.5
7×7	8.8	1.8	1.4	79.8
9×9	8.8	1.8	1.5	79.6
11×11	8.8	1.9	1.5	79.8

Table 2: Top-1 accuracy on ImageNet-1k dataset for FastViT-S12 model with varied kernel sizes in FFN and patch embedding layers.

Ablation	Description	Top-1 Acc.	Train Time (hrs)
No OverParam.	-	80.0	31.3
Train-Time OverParam.	Stem + Patch Emb.	80.6	33.4
	Stem + Patch Emb. + ConvFFN	80.6	40.1

Table 3: Train-time overparameterization ablation for FastViT-SA12 on ImageNet-1K. Extension to Table 3 in main paper. Train time is wall clock time elapsed at the end of a training run.

A.3. Training Time

We provide a coarse ablation of this in Table 3 of main paper. In our model, we do not overparameterize every element of the architecture, especially the parameter dense blocks like ConvFFN. In fact, we do not obtain any improvement by overparameterizing ConvFFN layers, verified empirically in Table 3. Since our model is partially overparameterized (only convolutional stem and patch embedding layers) during training, we do not see a significant degradation in train time as opposed to other train-time overparameterized models in literature which overparameterize all layers in a network. Note, all models were trained using the same hardware as described in Section 4.1 of main paper.

B. Experiments

B.1. Benchmarking

We follow the same protocol prescribed in [15] while benchmarking the model on an iPhone 12 Pro mobile device. To benchmark the models on desktop-grade GPU NVIDIA RTX-2080Ti, we first warmup the device by running the forward pass of TensorRT model for 100 iterations and then benchmark the model over a 100 iterations. We report the median latency value from 100 estimates. For image classification models we use a batchsize of 8 (similar approach was adopted in [7]) and a batchsize of 2 (due to GPU memory limits) for semantic segmentation and object detection models. Both mobile and GPU latency estimates can have a standard deviation of ± 0.1ms.

While benchmarking ConvNeXt [6] models on mobile device, we noticed the inefficiencies introduced by reshape ops causing increase in latency. While majority of hybrid models that use self-attention based token mixers require explicit reshaping of tensors. We can avoid this operation in ConvNeXt basic block by simply using a channel first implementation of LayerNorm and replacing `nn.Linear` layers with 1×1 convolutions. This simple change results in significant improvement in runtime as shown in Table 4.

Model	Mobile Latency (ms)	
	Before	After
ConvNeXt-T	33.5	3.7
ConvNeXt-S	66.4	5.4
ConvNeXt-B	89.1	8.4

Table 4: Benchmarking ConvNeXt before and after modifications.

Hyperparameter	Training T8, T12, S12, SA12, SA24, SA36, MA36	Fine-tuning SA36, MA36
Stochastic depth rate	[0.0, 0.0, 0.0, 0.1, 0.1, 0.2, 0.35]	[0.2, 0.4]
Input resolution	256×256	384×384
Data augmentation	RandAugment	RandAugment
Mixup α	0.8	0.8
CutMix α	1.0	1.0
Random erase prob.	0.25	0.25
Label smoothing	0.1	0.1
Train epochs	300	30
Warmup epochs	5	None
Batch size	1024	1024
Optimizer	AdamW	AdamW
Peak learning rate	1e-3	5e-6
LR. decay schedule	cosine	None
Weight decay rate	0.05	1e-8
Gradient clipping	None	None
EMA decay rate	0.9995	0.9995

Table 5: Training hyperparameters for ImageNet-1k experiments.

Model	Eval Image Size	Param (M)	FLOPs (G)	Mobile Latency (ms)	Top-1 Acc. (%)
MobileNetV3-S*[3]	224	2.5	0.06	**0.8**	67.4
MobileOne-S0[15]	224	2.1	0.3	**0.8**	71.4
MobileNetV2-x1.0[11]	224	3.4	0.3	0.9	72.0
DeiT-Ti[14]	224	5.7	1.3	4.8	72.2
MobileNeXt-x1.0[17]	224	3.4	0.3	0.9	74.0
EdgeViT-XXS[8]	224	4.1	0.6	1.4	74.4
MNASNet-A1[12]	224	3.9	0.3	1.0	75.2
FastViT-T8	256	3.6	0.7	**0.8**	**75.6**
MobileNetV2-x1.4[11]	224	6.9	0.6	1.4	74.7
MobileNetV3-L[3]	224	5.4	0.2	1.1	75.2
MobileNeXt-x1.4[17]	224	6.1	0.6	1.3	76.1
EfficientNet-B0[13]	224	5.3	0.4	1.7	77.1
EdgeViT-XS[8]	224	6.7	1.1	3.0	77.5
MobileOne-S3[15]	224	10.1	1.8	1.5	78.1
CMT-T*[2]	160	9.5	0.6	3.8	**79.1**
EfficientNet-B1[13]	256	7.8	0.7	2.5	**79.1**
FastViT-T12	256	6.8	1.4	**1.2**	**79.1**
MobileOne-S4[15]	224	14.8	2.9	1.9	79.4
DeiT-S[14]	224	22.0	4.6	5.3	**79.8**
FastViT-S12	256	8.8	1.8	**1.4**	**79.8**

Table 6: Comparison of different state-of-the-art Mobile architectures on ImageNet-1k classification. HardSwish is not well supported by Core ML, $*$ denotes we replace it with GELU for fair comparison.

B.2. Image Classification

We provide hyperparameters used for training models on ImageNet-1k dataset reported in Table 5 in main paper. Models are trained at resolution 256×256 and fine-tuned for 384×384. We follow the same training setup as [16, 14]. The hyperparameters used for all FastViT variants are listed in Table 5. For distillation experiments, we use RegNetY-16GF [9] as the teacher model similar to [14]. Additional hyperparameters are same as our image classification training procedure and are listed in Table 5. When trained using different seeds, results are within $\pm0.2\%$ in Top-1 accuracy.

B.3. Comparison with Mobile Architectures

We compare our model against highly efficient mobile architectures in Table 6 and in Figure 1. Our model outperforms recent state-of-the-art MobileOne [15] architecture which is purely convolutional. Our model also outperforms EdgeViT [8], which is a recent state-of-the-art light-weight ViT architecture.

B.4. Model Scaling

In this work, we sample architectures that are smaller than 50M parameters for efficient deployment. Similar to the Swin-T [5] variant, we use a stage compute ratio of 1:1:3:1 most of our variants and for the smallest variant we use 1:1:2:1. For our models, width doubles at each new

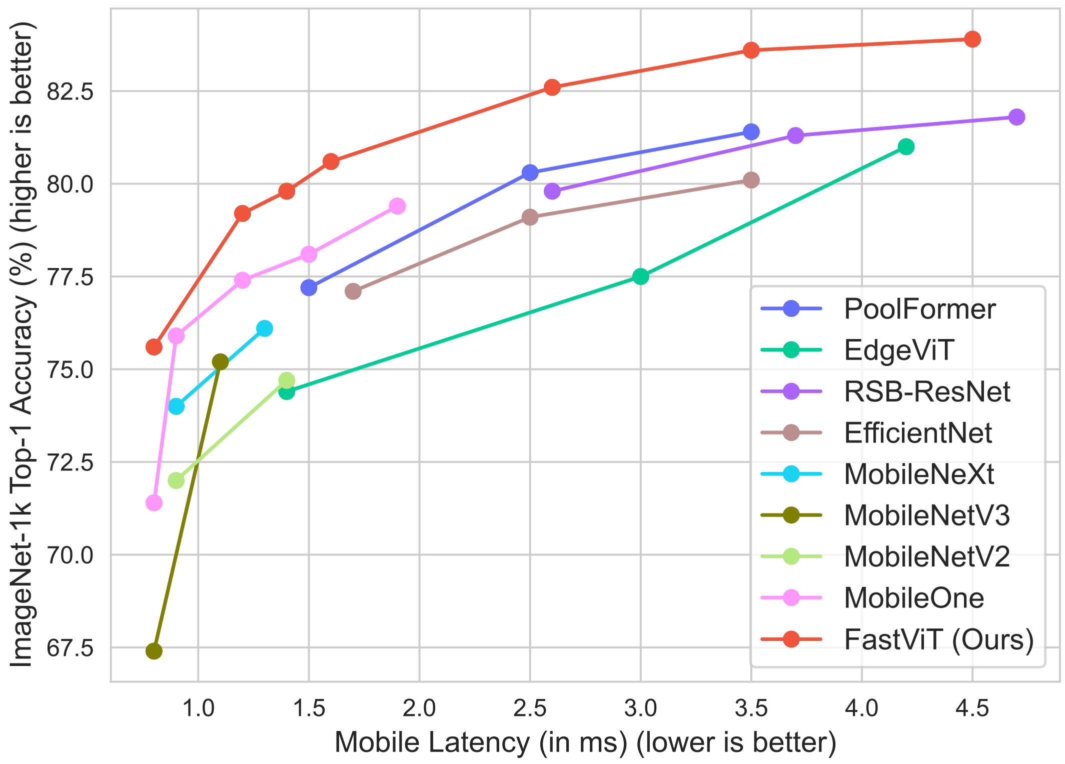

Figure 1: Accuracy vs. Mobile latency scaling curves of recent state-of-the-art Mobile Architectures and FastViT variants. The models are benchmarked on iPhone 12 Pro using the appropriate image sizes described in Table 6.

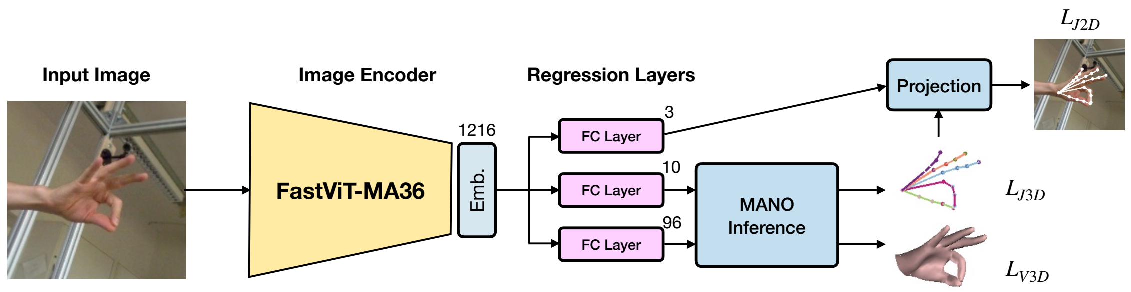

Figure 2: Overview of 3d hand mesh estimation framework.

stage. We use configurations of [48, 96, 192, 384], [64, 128, 256, 512] and [76, 152, 304, 608] in this paper. The tiny("T") variants use MLP expansion ratio of 3. Rest of the variants use an MLP expansion ratio of 4.

B.5. 3D Hand mesh estimation

Architecture As shown in Figure 2, our model uses simple regression layers to regress weak perspective camera, pose and shape parameters of the MANO model [10]. These layers are single fully connected layers unlike deep regression layers used in [1]. We regress 6D rotations [18] for all joints in MANO model. There are 3 losses to minimize in our framework, L_{V3D}, 3D vertex loss between predicted mesh and ground truth mesh. L_{J3D}, 3D joint loss between predicted 3D joints and ground truth 3D joints. L_{J2D}, 2D joint loss between projected 3D joints and ground 2D keypoints.

Setup We train our model on FreiHand [19] dataset, that contains 130,240 training images and 3,960 test images. Following METRO [4], we train our model on 224×224 images from the dataset using Adam optimizer. The models are trained for 200 epochs, with an initial learning rate of 1e-4 and decayed by a factor of 10 after 100 epochs. We initialize the backbone with weights obtained by pretraining on ImageNet-1k dataset. The weighting for all the losses in our framework is set to 1.0.

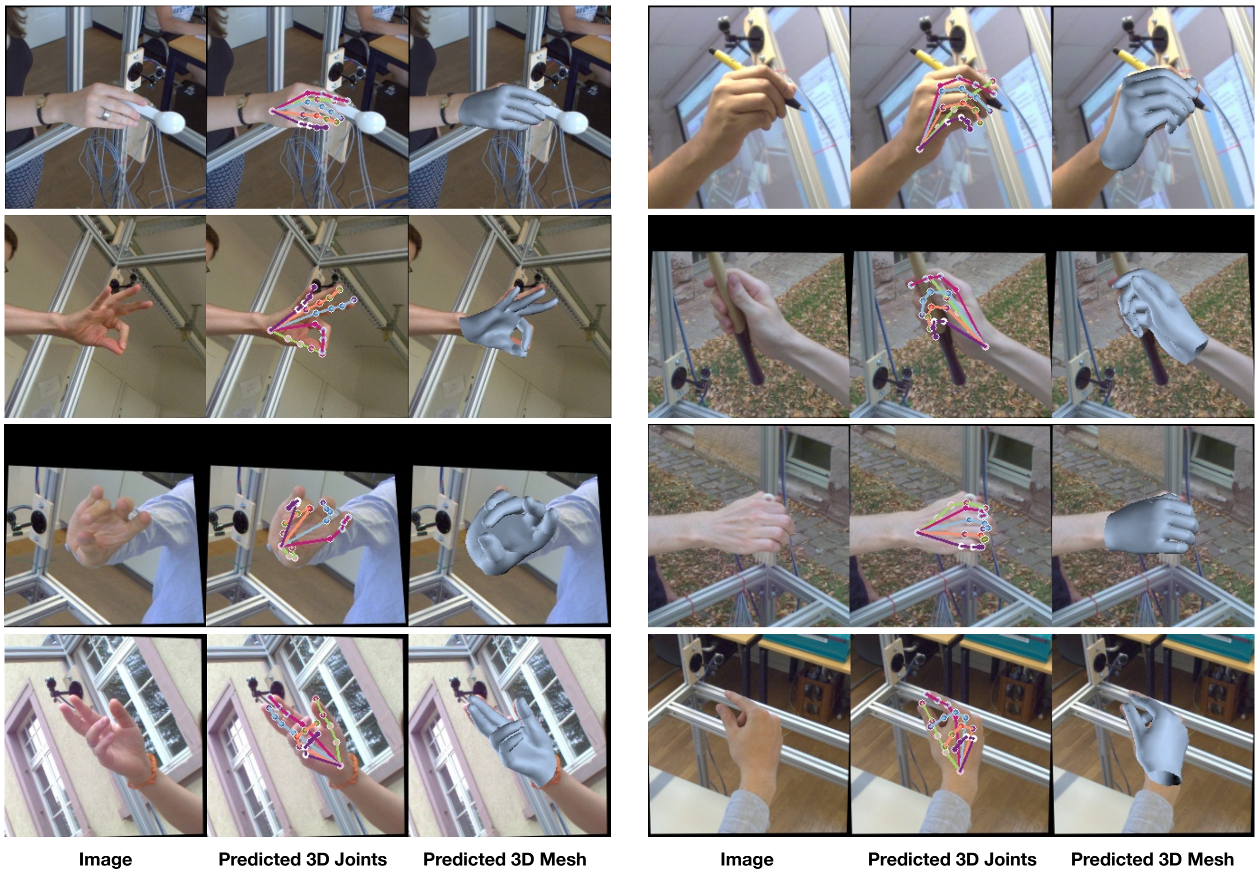

Figure 3: Qualitative results from our framework on FreiHand test set. 3D predictions are projected on to the image using weak perspective camera model, parameters for this camera model is also predicted by the model.

Results Figure 3 shows qualitative results from our framework on FreiHand test set. Even though our model is simple, it can model complicated gestures. Our model predicts reliable poses even in the presence of occlusion from hand-held objects.

References

[1] Lim Guan Ming, Jatesiktat Prayook, and Ang Wei Tech. Mobilehand: Real-time 3d hand shape and pose estimation from color image. In *27th International Conference on Neural Information Processing (ICONIP)*, 2020.

[2] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

[3] 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.

[4] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[5] 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. *arXiv preprint arXiv:2103.14030*, 2021.

[6] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[7] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision*, pages 116–131, 2018.

[8] 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*, 2022.

[9] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[10] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 2017.

[11] 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.

[12] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

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

[15] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. An improved one millisecond mobile backbone. *arXiv preprint arXiv:2206.04040*, 2022.

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

[17] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[18] Yi Zhou, Connelly Barnes, Lu Jingwan, Yang Jimei, and Li Hao. On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[19] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 813–822, 2019.