Which Transformer to Favor: A Comparative Analysis of Efficiency in Vision Transformers -Supplementary Material-

Tobias Christian Nauen[1,2]
tobias_christian.nauen@dfki.de

Sebastian Palacio[3]
sebastian.palacio@de.abb.com

Federico Raue[2]
federico.raue@dfki.de

Andreas Dengel[1,2]
andreas.dengel@dfki.de

[1]University of Kaiserslautern-Landau, Gottlieb-Daimler-Straße, Kaiserslautern
[2]German Research Center for Artificial Intelligence (DFKI), Trippstadter Straße 122, Kaiserslautern
[3]ABB AG, Kallstadter Str. 1, 68309 Mannheim

Abstract

This is the supplementary material of the paper 'Which Transformer to Favor: A Comparative Analysis of Efficiency in Vision Transformers'.

A. Implementational Details

A.1. Efficient Transformers

Our implementation is based on the `PyTorch` framework [8] and the `timm` package [13]. Whenever possible, we utilize the original implementations provided by the authors of each transformer variant. In cases where PyTorch implementations were not available, we adopt other existing open-source implementations or develop our own as needed. Specific sources of each implementation can be found in the corresponding code files or the README file[1]. We make minimal modifications to these implementations to ensure consistent handling of input and output sizes and for the accommodation of additional loss terms. For models that follow the overarching structure of ViT [3], we only change the implementation of the attention mechanism, keeping the rest the same as ViT.

A.2. Training Pipeline Details

The training pipeline is taken from *DeiT III* [11], building upon the well-established pipeline outlined in *DeiT* [10]. Table 1 presents the default hyperparameters and training pipeline details. For the non-default hyperparameters used in specific training runs, please consult Table 6. Our training process involves initial pretraining at resolutions of 192×192 and 224×224 pixels, followed by fine-tuning at 224×224 pixels. The models pretrained at 224×224 pixels additionally get finetuned at 384×384 pixels, as long as this fits onto 8 NVIDIA-A100s (80 GB). In cases where training (especially pretraining) is unstable, we adjust some hyperparameters in accordance with the original papers, beginning with the learning rate, warmup epochs, and weight decay.

A.3. Efficiency Evaluation

For calculating the theoretical metrics, we directly count the number of parameters of the PyTorch implementation by iterating over `model.parameters()`. We count the number of floating-point operations using the `fvcore` package[2], with additional handles for counting FLOPs from operations beyond matrix multiplication.

[1]https://github.com/tobna/WhatTransformerToFavor
[2]https://github.com/facebookresearch/fvcore/tree/main

Table 1. Training pipeline details and hyperparameters. In 3-Augment, we always choose one of grayscale, solarize, or Gaussian blur at random.

	Pretrain	Finetune
Dataset	ImageNet-21k [9]	ImageNet-1k [2]
Epochs	90	50
LR	3×10^{-3}	3×10^{-4}
Schedule	cosine decay	
Min LR	10^{-5}	
Batch Size	2048	
Warmup	linear	
Warmup Epochs	5	
Warmup LR	10^{-6}	
Weight decay	0.02	
Grad Clipping	1.0	
Label Smoothing	0.1	
Drop Path Rate	0.05	
Optimizer	Lamb [16]	
Optimizer ε	10^{-7}	
LayerScale Init	10^{-4}	
Dropout prob.	0.0	
Mixed precision	AMP	
Augmentation	3-Augment [11]	
Flip prob.	0.5	
Crop	Simple Random Crop	
Resize to	224×224 or 192×192	224×224 or 384×384
Grayscale prob.	0.33	
Solarize prob.	0.33	
Gauss. Blur prob.	0.33	
Color Jitter Fact.	0.3	
Cutmix prob.	1.0	
Normalization	$\mu = (0.485, 0.456, 0.406)$ $\sigma = (0.229, 0.224, 0.225)$	
GPUs	4 (or 8) NVIDIA A100	

Memory requirements, both at training and inference time, are measured using `torch.cuda.max_memory_allocated()`. To measure training time, we capture differences of Python's `time.time` across each training epoch. Throughput is measured using CUDA Events[3] and averaged over 1000 iterations. This is additionally done at multiple batch sizes, including powers of two, and others close to the GPUs VRAM limit. We report the best result together with the corresponding batch size.

A.4. Variability of Efficiency Metrics

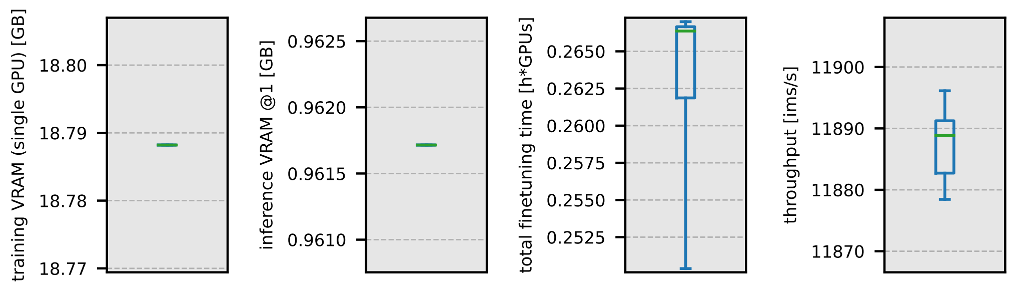

Figure 1. Box plots of efficiency metric ranges over 8 independent training runs.

[3] https://pytorch.org/docs/stable/generated/torch.cuda.Event.html

To validate the reliability of our measurements of these metrics, we inspect each metric's variability by empirically assessing it for the baseline of ViT. Figure 1 shows ranges of the empirical efficiency metrics across 8 training runs. Particularly, we observe negligible variability in the training and inference VRAM measurements, with only marginal variability in time-based metrics. For instance, the range of throughput measured is 11878 to 11896 images per second, equivalent to 0.15% of the mean, with a standard deviation of 6.48, 0.05% of the mean. Finetuning time measurements have a slightly larger range from 0.2504 to 0.2670 GPU-hours per epoch, or 6.33% of the mean, with a standard deviation of 0.0075, which represents 2.84% of the mean. We attribute this marginal increase in variation predominantly to external factors, such as server temperature and load, as well as nuanced dissimilarities between specific instances of the same hardware. Overall, the measurements of all efficiency metrics are very robust.

B. Model Selection

We systematically select efficient transformers based on diversity, popularity, and novelty of their approaches. These models, from the domains of CV and NLP, all claim to improve the efficiency of the baseline transformer. We select papers that are listed under the term *efficient transformer*[4] and are either published since 2018 with atleast 50 citations or published since 2023 with atleast 10 citations. We also select papers that are listed under *efficient ViT*[5] since 2018 with atleast 5 citations. Additionally, we include papers from recent high-ranking conferences, like NeurIPS 2023, AAAI 2024, ICML 2024, ICLR 2024, and WACV 2024. We then add models that divise unusual approaches to make transformers more efficient, like the *Switch Transformer* or *FNet*, to increase the diversity of the benchmarked strategies. We only keep papers that introduce changes to the transformer architecture in order to make it more efficient and have also published their code[6]. We exclude models that have architectures specifically designed for one specific task only, like semantic segmentation, point clouds, or spherical images.

C. Other Datasets & Tasks

Table 2. Hyperparameter changes for finetuning models on other datasets. All the other parameters are the same as in Table 1.

Parameter	Cars	Flowers	Places
epochs	2000	2000	50
lr	3e-4	3e-4	3e-4
batch size	1024	256	2048
resize to	224	224	224

We evaluate on image classification, as it is considered a challenging yet well-studied task to compare models. However, we think that extending the benchmark to object detection and segmentation will give additional insights about efficiency in vision transformers. We find this extension to be non-trivial, because of the added complexity of introducing a patch/token-level to pixel-level decoder and because some models remove/change tokens from the sequence, losing local information. Therefore, evaluating on dense task falls out of the scope of our paper, and we consider it future work.

Additionally to ImageNet, we further finetune models on the Stanford Cars [6], Oxford Flowers 102 [7], and MIT Places365 [17] datasets using the hyperparameters in Table 2. Model accuracies for these down-stream tasks are presented in Table 3. We observe a high correlation of accuracies between the down-stream and ImageNet performances, with even higher rank-correlations meaning that the order of models does *not* significantly change from one dataset to another. This observation is corroborated by Figure 2 where we plot the Pareto front of throughput and accuracy for the four different datasets. Even though the distribution of some models changes, as they are finetuned on other tasks, the Pareto front stays largely the same, with differences probably stemming from the different number of training examples in the datasets.

[4]https://www.semanticscholar.org/search?year[0]=2018&year[1]=2024&q=efficient%20transformer&sort=total-citations

[5]https://www.semanticscholar.org/search?year[0]=2018&year[1]=2024&q=efficient%20ViT&sort=total-citations

[6]Full PyTorch implementation needed to be published until the 1st of July 2024.

Table 3. Model performance on different datasets and correlation coefficients of datasets.

Model	Taxonomy Class	ImageNet-1k	Cars	Flowers	Places
ViT-S/16	Baseline	82.54	91.58	98.87	58.56
ViT-Ti/16	Baseline	74.27	90.32	96.66	55.14
ViT-B/16	Baseline	84.80	92.00	99.53	59.13
DeiT-S/16	Baseline	82.29	92.24	98.89	58.08
ResNet50	Baseline (ResNet)	77.90	92.63	96.88	56.06
Nystrom ViT-64-S/16	Low-Rank Attention	80.83	89.71	98.70	57.87
Nystrom ViT-32-S/16	Low-Rank Attention	80.54	87.96	98.47	57.53
Linformer-S/16	Low-Rank Attention	81.46	90.37	98.66	58.11
XCiT-S-12/16	Low-Rank Attention	83.65	93.27	99.30	58.71
SwinV2-Ti-W7/4	Sparse Attention	83.11	92.21	98.72	58.46
Swin-Ti-W7/4	Sparse Attention	82.91	92.89	99.08	58.48
Swin-S-W7/4	Sparse Attention	84.87	92.74	99.56	59.02
Sinkhorn Cait-S-Bmax-32/16	Sparse Attention	81.27	92.79	98.48	57.64
Routing ViT-S/16	Sparse Attention	73.92	84.06	97.37	55.44
WaveViT-S	Sparse Attention	83.91	92.69	99.33	57.72
Informer-S/16	Sparse Attention	73.47	83.69	96.73	52.50
Synthesizer FD-S/16	Fixed Attention	75.51	87.74	96.69	56.02
Synthesizer FR-S/16	Fixed Attention	79.07	91.16	98.30	57.14
PolySA-S/16	Kernel Attention	78.44	88.66	97.05	56.82
SLAB-S/16	Kernel Attention	78.70	91.10	98.55	56.42
Hydra-S/16	Kernel Attention	77.84	86.96	97.22	56.71
EfficientFormerV2-S0	Hybrid Attention	71.53	89.51	95.55	53.43
CvT-13	Hybrid Attention	82.35	90.72	98.70	58.17
CoaT-Ti	Hybrid Attention	78.42	90.45		55.41
Efficient ViT-B-2	Hybrid Attention	81.52	88.29	97.36	56.85
NextViT-S	Hybrid Attention	83.92	92.90	98.89	58.28
ResT-S	Hybrid Attention	79.92	92.23	98.72	55.71
Mixer-S/16	Non-Attention Shuffling	76.20	86.61	96.72	55.66
FocalNet-S-Srf	Non-Attention Shuffling	84.91	93.13	99.48	58.87
FocalNet-Ti-Srf	Non-Attention Shuffling	83.23	93.29	99.08	58.59
SwiftFormer	Non-Attention Shuffling	76.41	90.69	96.61	55.20
FastViT-12	Non-Attention Shuffling	78.77	92.04	96.98	56.72
Efficient Mod-S	Non-Attention Shuffling	80.21	92.01	98.09	57.33
FNet-S/16	Fourier Attention	73.12	87.87	95.52	54.64
GFNet-S/16	Fourier Attention	81.33	91.92	98.80	57.97
AFNO-S/16	Fourier Attention	78.55	90.30		56.09
AFNO-Ti/16	Fourier Attention	68.87			
EViT-S/16	Token Removal	82.29	90.97	98.86	58.39
Dynamic ViT-S/16	Token Removal	81.09	91.48	98.39	58.00
EViT Fuse-S/16	Token Merging	82.20	90.53	98.75	
ToMe-S-R-8/16	Token Merging	82.11	91.58	98.61	58.20
Token Learner-8-75-S/16	Summary Tokens	80.69	91.66	97.34	56.40
Token Learner-8-50-S/16	Summary Tokens	77.35	90.29	94.56	54.37
STViT-Swin-Ti-W-7/4	Summary Tokens	82.22	91.37	98.58	58.45
CaiT-S-24	Summary Tokens	84.91	92.94	99.50	53.48
Switch ViT-8-S/16	More MLPs	82.39	89.14	98.97	58.22
Hi ViT-Ti/16	More MLPs	76.13	91.13	97.25	
Correlation	with ImageNet-1k	1.0	0.70	0.89	0.80
Spearman's ρ	with ImageNet-1k	1.0	0.71	0.93	0.83

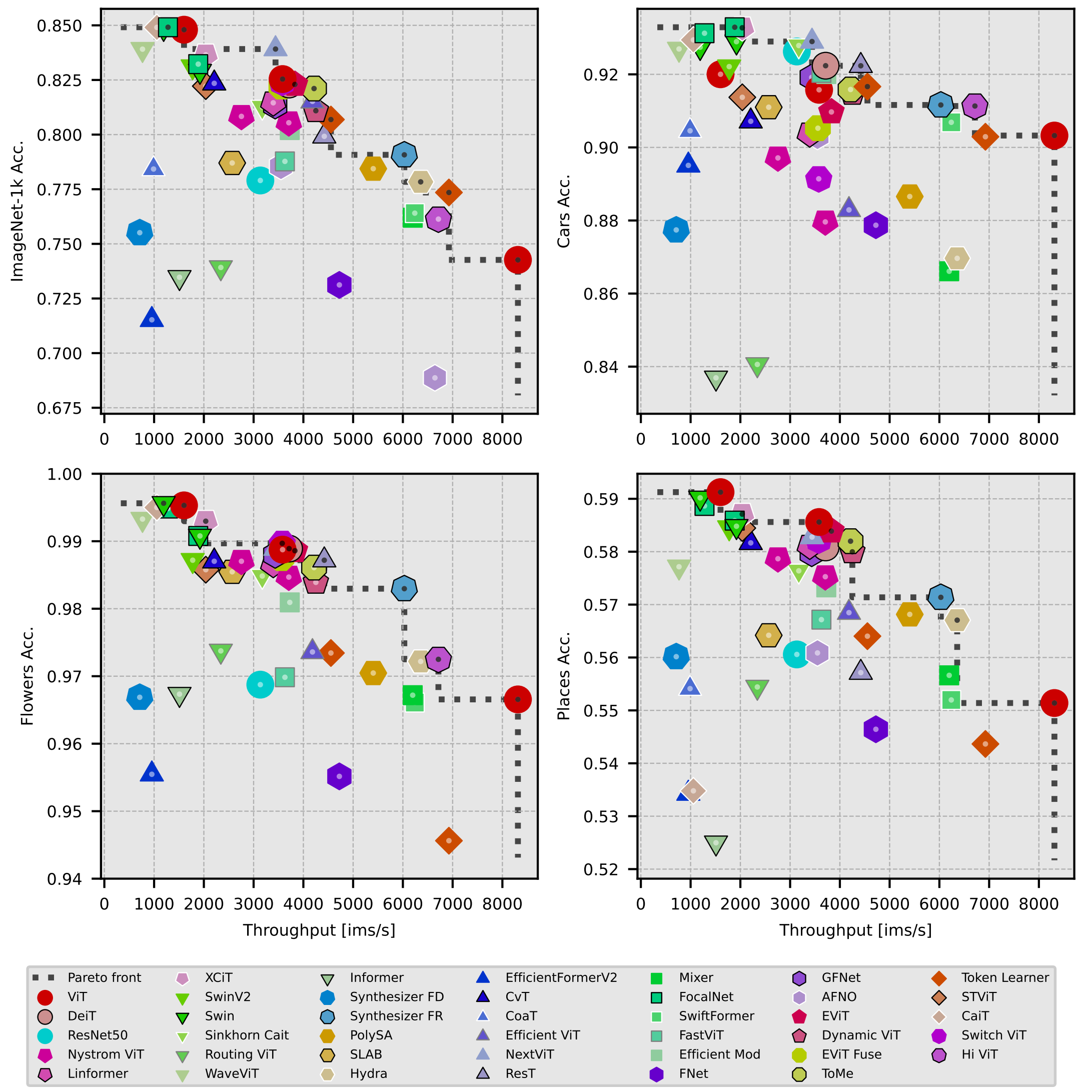

Figure 2. Pareto front of throughput and accuracy on four datasets for images of size 224×224 px. While the distribution of some models changes, the Pareto fronts are always very similar, independent of the dataset.

D. Results: Details

D.1. Training Failures

Despite our efforts to hyperparameter tune the models based on their original papers, we encounter training failures with several models using the new pipeline. Specifically, we are unable to achieve convergence with the *Performer* [1], *Linear Transformer* [4], and *HaloNet* [12] models.

The *Performer* and *Linear Transformer* are found to be consistently unstable across all hyperparameter configurations, despite adjusting various training parameters such as learning rate, weight decay, and batch size. In the case of *HaloNet*, we find the implementation[7] to be prohibitively slow, and experiments indicate that the model does not exhibit meaningful learning even after multiple epochs of training on ImageNet-21k with different hyperparameters. Specifically, after more than 10 epochs of training, the *HaloNet* model is still performing no better than random choice, and due to the high computational cost of training, we do not pursue further hyperparameter tuning. The *Reformer* [5] model needs too much memory to fit on 8 NVIDIA A100s (40GB), and we therefore did not train it. Additionally, we only achieve convergence with *A-ViT* [15] using a very small learning rate, which results in suboptimal results within our fixed number of epochs. We also don't reach satisfying results with *GTP* [14] and therefore exclude it from our analysis. Overall, we find it harder to get the kernel attention and some of the more complicated sequence reduction models to converge, than models of the other categories.

We note that these results may not be representative of the performance of these models, and further investigation is needed to understand the causes of the training failures.

D.2. ImageNet Accuracy in Taxonomy Classes

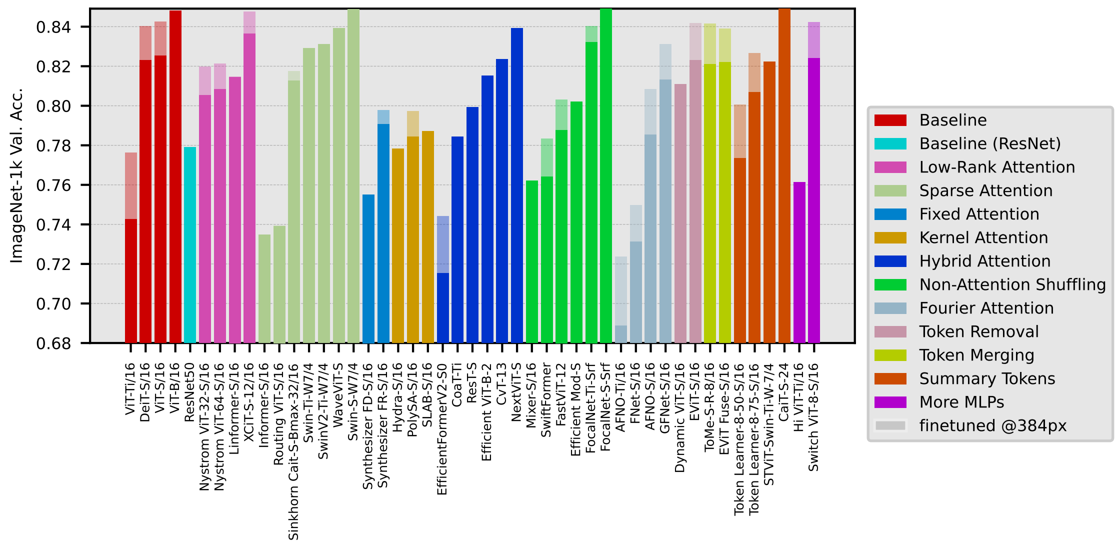

Figure 3. Accuracy of trained models trained, grouped by taxonomy classes and ordered by accuracy inside the classes. The base bar is the model trained at 224×224 pixels, while the lighter bar is the same model fine-tuned at 384×384 pixels, if training fits on 8 NVIDIA A100s and the accuracy is higher than the base model.

Figure 3 shows the ImageNet accuracy achieved by the trained models. The models are categorized into distinct taxonomy classes. Notably, we observe that while models belonging to various taxonomy classes reach the highest accuracy levels, those falling under the *fixed attention* and *kernel attention* categories appear to demonstrate comparatively lower accuracy outcomes. For *fixed attention*, we hypothesize this may be due to the process of using the same attention matrix for every input image, while for *kernel attention* it might be due to convergence difficulties or numerical problems during training.

D.3. Speed on Different Hardware

We extend our analysis beyond large data center GPUs to CPUs and consumer GPUs. Table 4 shows a strong correlation between all the architectures with Spearman correlations exceeding 75%. While data center GPU and consumer GPU

[7] based on the halonet_pytorch package https://github.com/lucidrains/halonet-pytorch

Table 4. Correlation coefficient (left) and Spearman correlation (right) of measuring speed on different hardware – NVIDIA A100 (A100), NVIDIA GeForce RTX 3090 (RTX 3090), and Intel i9-12900H (CPU).

Correlation	CPU	RTX 3090
A100	0.49	0.97
CPU		0.75

Spearman ρ	CPU	RTX 3090
A100	0.75	0.96
CPU		0.80

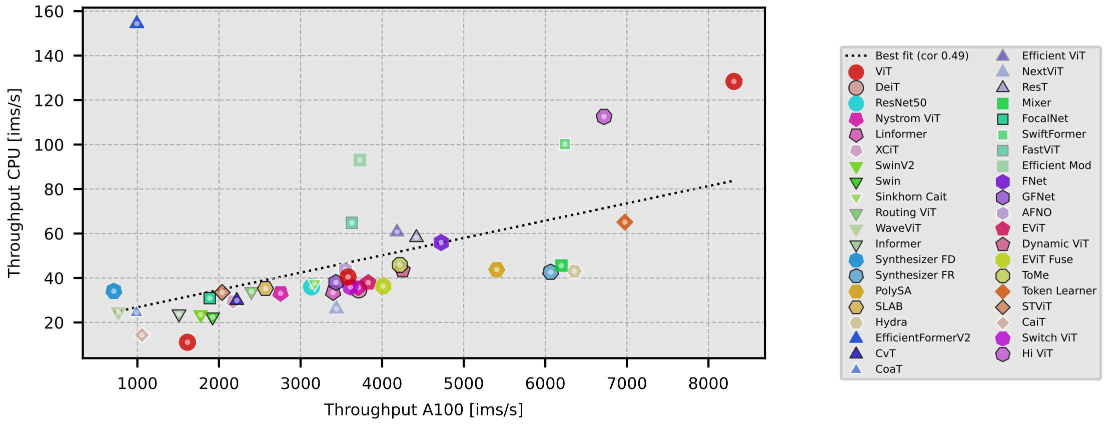

Figure 4. Throughput comparison on NVIDIA A100 and Intel i9-12900H (CPU) with line of best fit. Most models closely follow a linear relationship, with two very notable outliers. In particular *EfficientFormerV2* [?] is one of the slowest models on GPU, but one of the fastest ones on CPU.

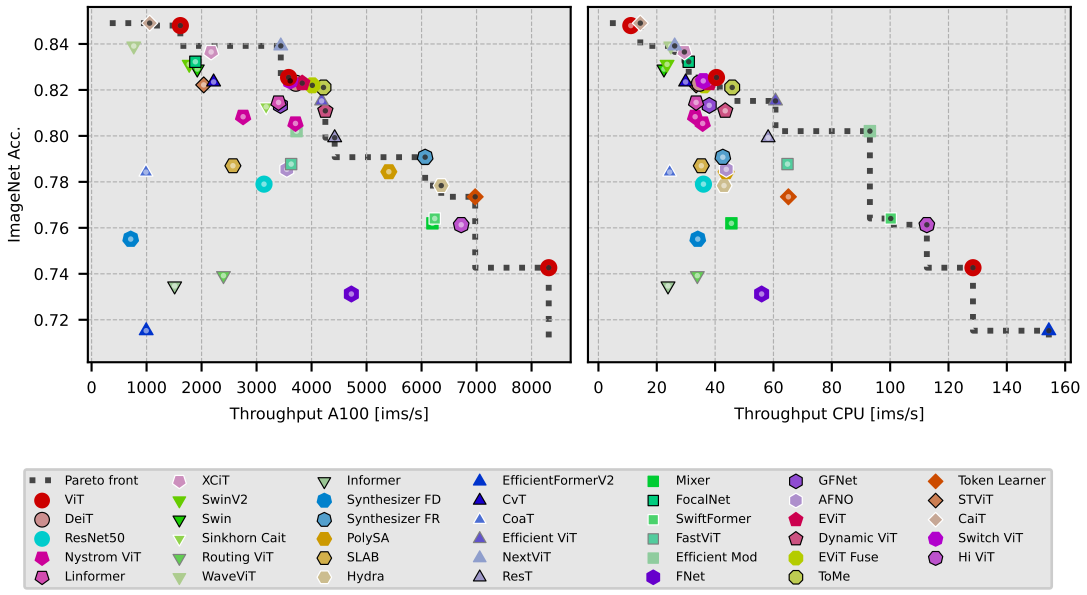

Figure 5. Pareto front of accuracy (ImageNet-1k) and throughput on NVIDIA A100 and Intel i9-12900H (CPU). We observe that the Pareto fronts are largely the same, with the one on CPU looking more squished because of the relatively larger speed of ViT-Ti, which we already observed in Figure 4 to be an outlier.

performance are 97% correlated, GPU-CPU correlation is lower, with Spearman correlation between 75% and 80%. When plotting GPU-throughput against CPU-throughput in Figure 4, we observe most models adhering to a linear relationship with some notable outliers: *EfficientFormerV2*, *HiViT*, *SwiftFormer*, *EfficientMod* and *ViT-Ti*. These models demonstrate significantly faster performance on CPU relative to GPU, while *ViT-B* exhibits the opposite trend, being notably slower on CPU compared to GPU. We hypothesize that these disparities stem from differences in memory access patterns between GPUs and CPUs. While some very small models are faster on CPU than predicted by GPU performance, larger models get slower. The presence of these outliers reduces the overall correlation to 49%. Without outliers, the correlation for the remaining models becomes 76%. A similar picture paints itself in Figure 5, where the overall Pareto front on CPU looks similar to the one on GPU, but a little squished because of the outstanding CPU-speed of *ViT-Ti* and *EfficientFormer*. Again, we find *ViT-S* and *ViT-Ti* at the Pareto front, but not *ViT-B*, due to the different scaling behavior on CPU. Additionally, we find some of the outliers from before on the Pareto front. Namely, *EfficientFormerV2*, *SwiftFormer*, *HiViT*, and *EfficientMod* are on the Pareto front on CPU, but not on GPU. We again attribute this to different memory access properties on GPU and CPU, as *Synthesizer* has a constant, learned attention matrix stored in memory.

D.4. Model Scaling & Image Scaling

In Figure 6 we show examples for the different effects of scaling the model size vs. scaling the image resolution. For the three combinations of models *ViT-Ti* $\rightarrow$ *ViT-S*, *ViT-S* $\rightarrow$ *ViT-B*, *FocalNet-Ti* $\rightarrow$ *FocalNet-S*, and *AFNO-Ti* $\rightarrow$ *AFNO-S* (one per column) we plot the effects of scaling the image resolution at the smaller model size and scaling the model, but still using the smaller image size on the tradeoff of accuracy and one of the empirical efficiency metrics. We observe that scaling the model size is almost always advantageous compared to scaling the image resolution. The only exception being inference memory, where scaling the image resolution seems to work well for small models. This is similar to the overall trend using larger images when optimizing for inference memory, that we found in the main paper. Note, that in Figure 1 of the main paper, where we plot the Pareto front of all models and the Pareto front when only considering the models using high resolution images, it looks like both fronts are the same for high accuracy models. This might be due to the fact that we did not include even larger models and therefore can not extend the Pareto fronts for even higher accuracy models. In particular, the Pareto front in out plots goes flat at the top, where we do not have any more models to extend it.

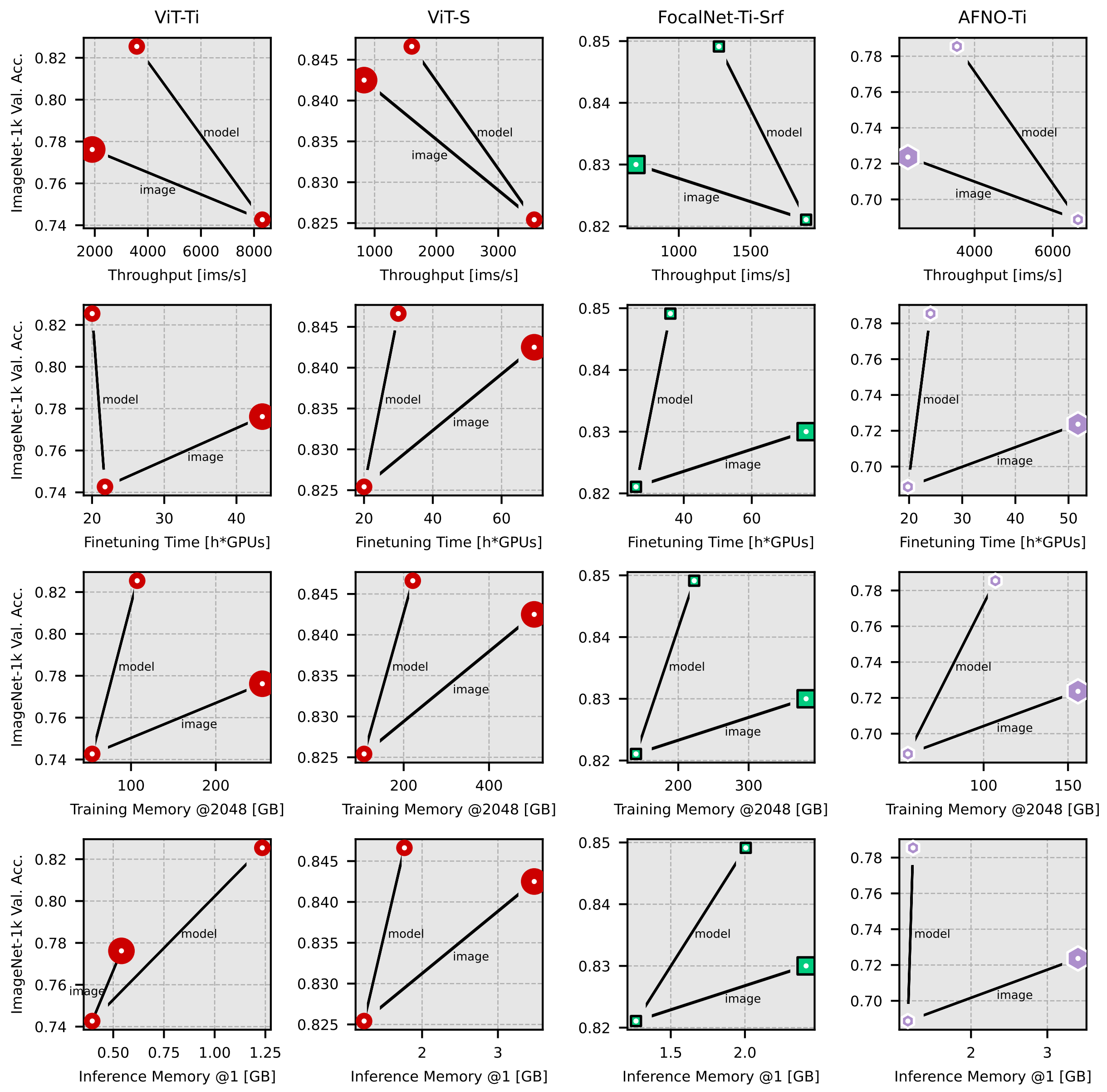

Figure 6. Effects of scaling the model size and the image resolution. Small markers use 224px images, while large markers use 384px images. Columns are different model combinations with titles marking the smaller model. For scaling the model, we use the next larger size. The rows are different dimensions of efficiency on the x-axes. We measure training memory at our default batch size of 2048 and inference memory at the minimum possible batch size of 1.

D.5. Evolution of Models over Time

Figure 7 shows the Pareto front of throughput and accuracy over time. We observe that the Pareto front has not changed significantly since 2020, with only marginal improvements in throughput and accuracy. In 2021, the TokenLearner was added and in 2022 Hydra, EViT, and ToMe were added to the Pareto front.

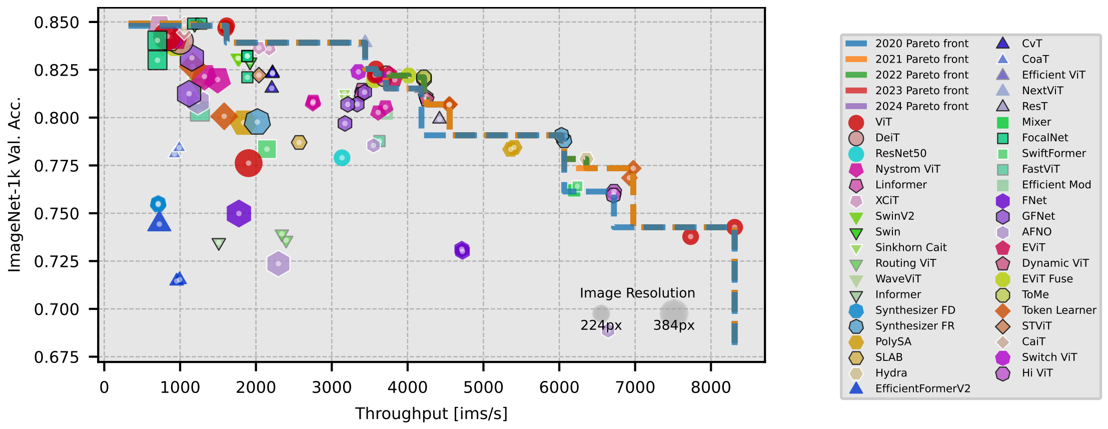

Figure 7. Pareto front of throughput and accuracy over time. Each fronts includes all models that were published up to that year. The improvements after 2020 are only marginal.

D.6. Pretraining Image Resolution

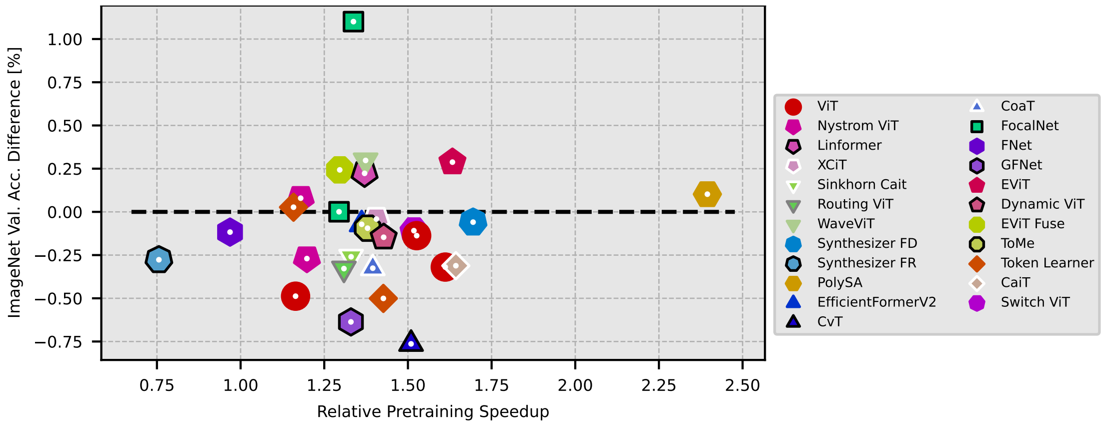

Figure 8. Impact of pretraining on ImageNet-21k at a lower resolution of 192px instead of 224px. Relative pretraining speedup is the ratio of pretraining time with 224px images and 192px images. All models are finetuned on ImageNet-1k at 224px and we plot the difference of finetuned accuracy.

When pretraining at a lower resolution of 192px images, we find an average decrease in the accuracy of 0.16% after finetuning on ImageNet-1k at a resolution of 224px for a 1.39 times pretraining speedup on average. For most models, the speedup is between 1.1 times and 1.6 times. While accuracy increases for some models, for most it decreases with up to 0.8% accuracy loss on ImageNet-1k.

D.7. Accuracy per Parameter

Figure 9 shows the full list of models ordered by accuracy and the accuracy per parameter. In the main paper, Synthesizer FR-S/16 to EViT-S/16 are collapsed into the *others* bar, as their accuracy per parameter is very similar.

Figure 10 shows the Pareto front of plotting the number of parameters against accuracy. In this metric, using a larger image size of 384 does not show up and therefore is an advantageous strategy to increase accuracy. Notably, the only Pareto optimal model that uses the smaller image size is CoaT-Ti.

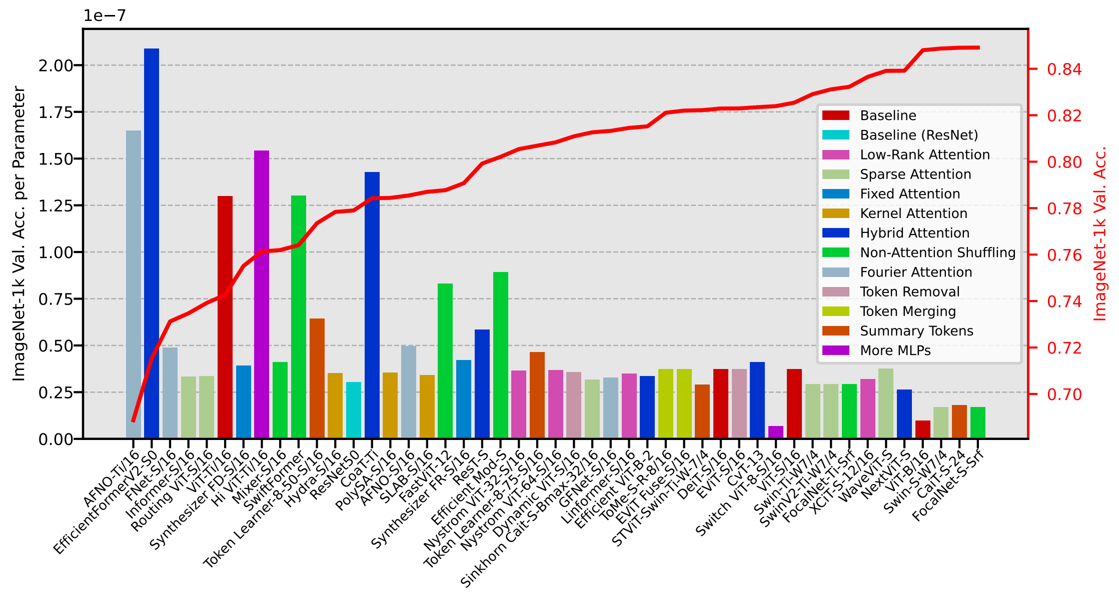

Figure 9. Accuracy (red line, right y-axis) and accuracy per parameter (bars, left y-axis) of different models at a resolution of 224×224 pixels. Models are ordered by accuracy.

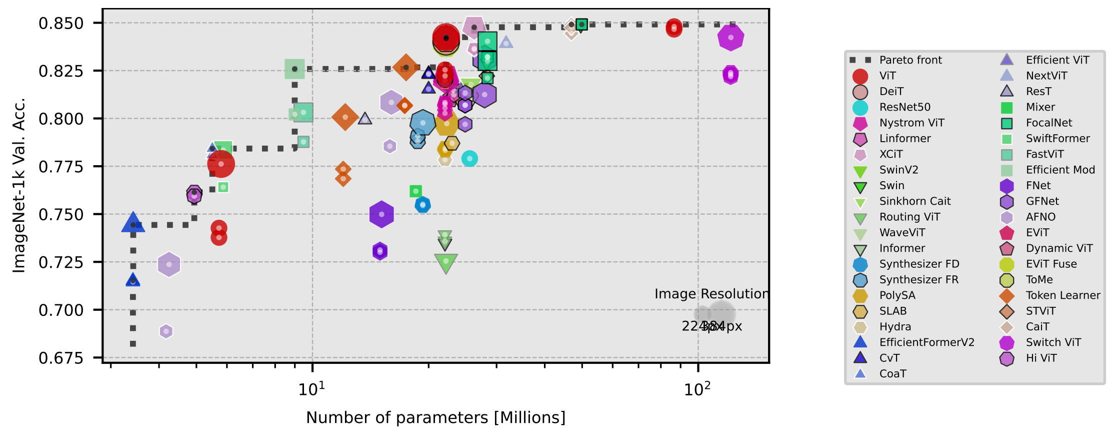

Figure 10. Pareto front of parameters and accuracy. We have a logarithmic x-axis in order to better show the differences between models. Since scaling to a larger image size is not represented in this metric, most Pareto optimal models use the larger image size.

D.8. Fine-tuning Time

Figure 11 visualizes the correlation of 0.8 between fine-tuning time and inference time. Except the few outliers at the top, this is especially pronounced for fast models and scatters more and more moving towards the slower ones.

Furthermore, Figure 12 shows the Pareto front of finetuning time and accuracy, which is similar to the one on inference time and accuracy. This similarity can be explained by the rather high correlation of finetuning time and inference time, we observed above.

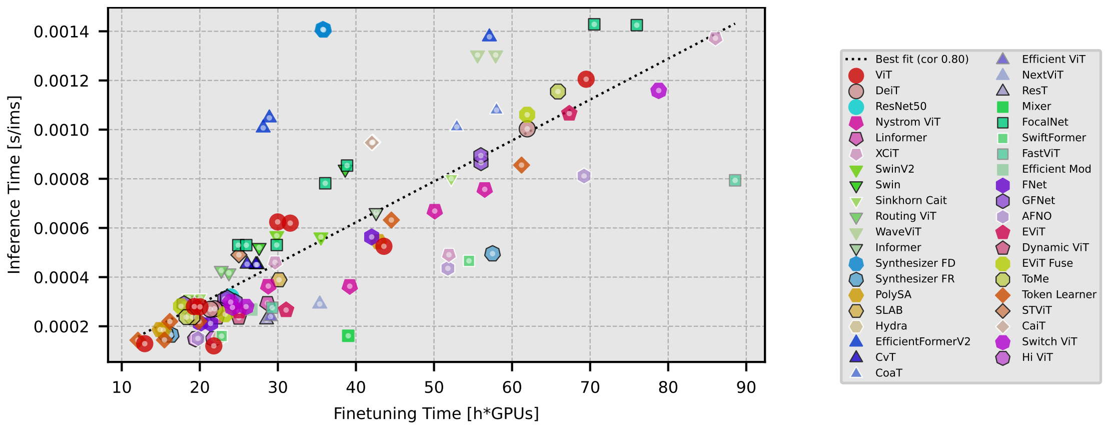

Figure 11. Fine-tuning time vs. inference time. This includes all runs at resolutions 224 × 224 and 384 × 384 pixels.

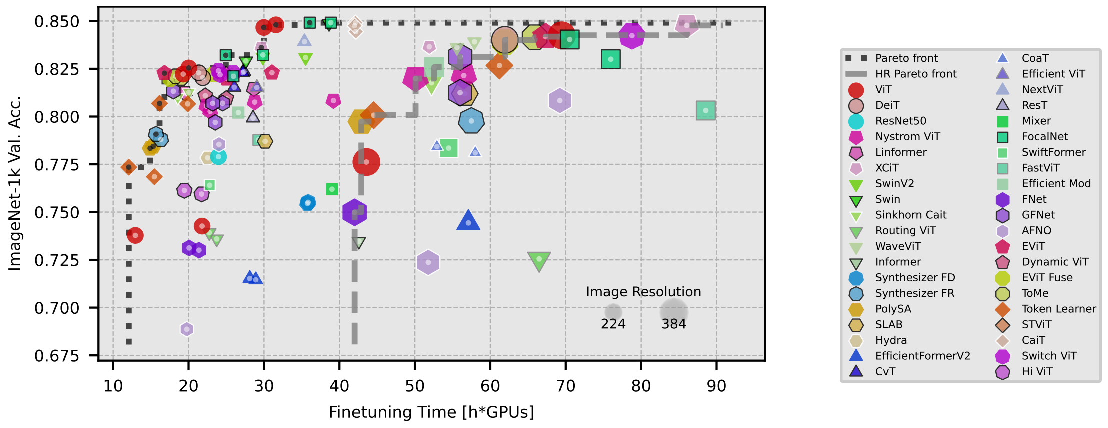

Figure 12. Pareto front of fine-tuning time vs. accuracy.

D.9. Training Memory

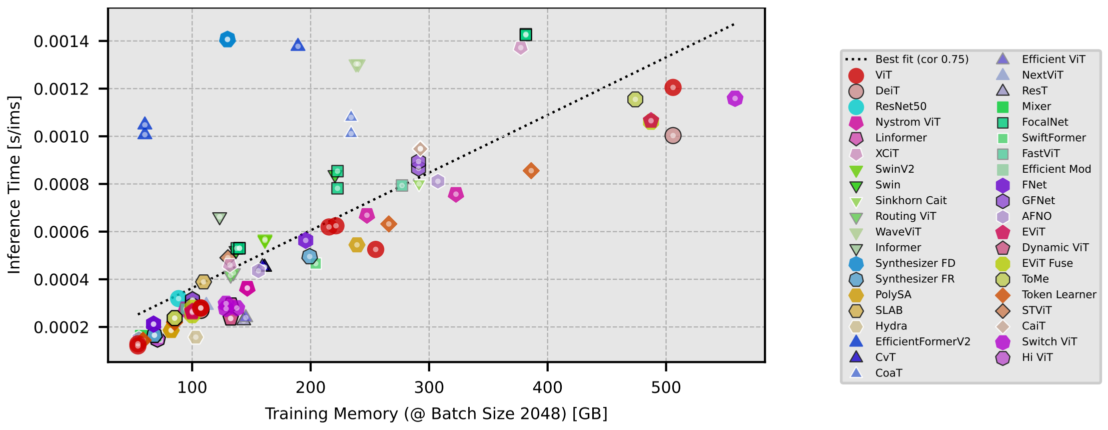

Figure 13. Training memory vs. inference time.

Figure 13 visualizes the correlation of 0.75 between training memory and inference time. This makes the Pareto front of training memory similar to the one of throughput. Nonetheless, we observe some significant outliers, that need relatively little memory for training, but are slow.

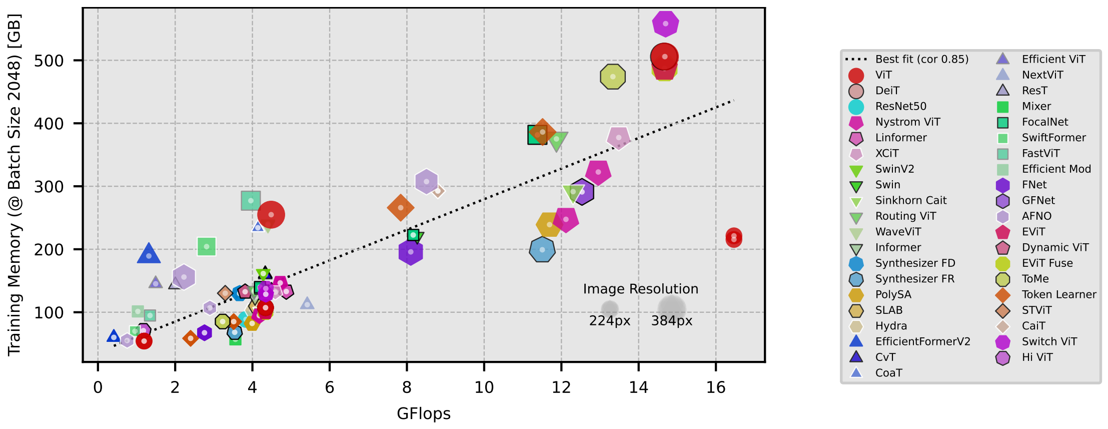

Figure 14. Flops vs. training memory.

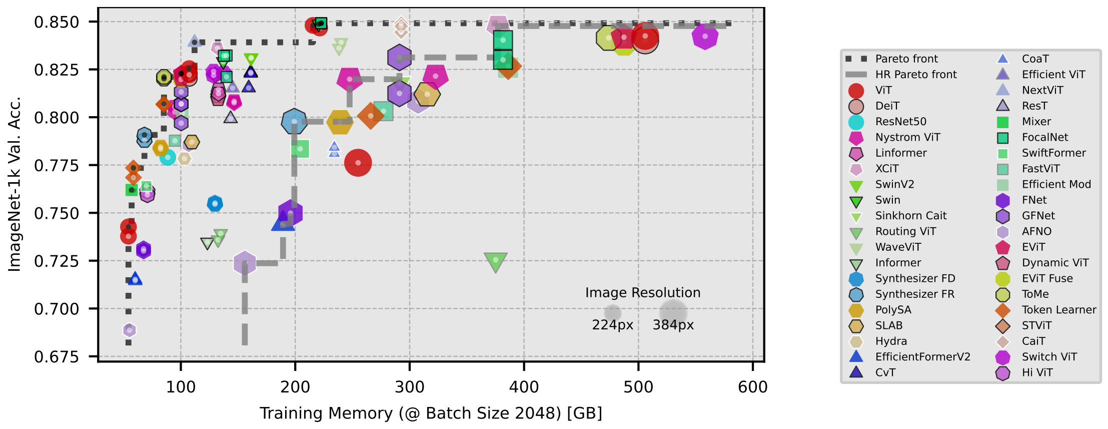

Figure 15. Full plot of training memory against accuracy. In the main paper, we only show the excerpt of $\leq$ 225GB training memory.

Figure 14 plots Flops against training memory of training runs at multiple resolutions. A robust correlation coefficient of 0.85 spans this diverse array of scenarios, enabling us to leverage the theoretical metric of throughput for an estimation of VRAM requirements during training, using the equation

$$\text{VRAM [GB]} \approx 25.43 \cdot \text{GFlops} + 25.50.$$

There are no significant outliers, similar to the level observed in Figure 13, to this relationship.

Figure 15 shows the full plot of training memory against accuracy. In the main paper, we only include an excerpt.

D.10. Pareto Optimal Models

Table 5 presents a comprehensive list of all Pareto optimal models for speed and memory consumption across both training and inference phases, grouped by taxonomy class. We highlight select models from this list in the flowchart featured in the main paper.

Table 5. List of Pareto optimal models for speed and memory consumption and training and inference. Models are grouped by taxonomy class. Models annotated with ↑ are using 384px images.

	Number of	Speed		Memory	
	Architectures	Inference	Training	Inference	Training
Baseline	3	ViT	ViT		ViT
Low-Rank Attention	3		XCiT		
Sparse Attention	6				Swin
Fixed Attention	2	Synthesizer-FR	Synthesizer-FR		Synthesizer-FR
Kernel Attention	3	Hydra	PolySA		
Hybrid Attention	6	NextViT		CoaT EfficientViT CvT NextViT	NextViT
Fourier Attention	3				
Non-Attention Shuffling	5	FocalNet	FocalNet	EfficientMod	Mixer FocalNet
Token Removal	2	EViT DynamicViT	EViT	EViT@384	EViT
Token Merging	2	EViT Fuse ToMe			ToMe
Summary Tokens	3	TokenLearner	TokenLearner	CaiT	TokenLearner
More MLPs	2				

References

[1] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. 6

[2] 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*. IEEE, 2009. 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1

[4] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *International Conference on Machine Learning*, pages 5156–5165, 2020. 6

[5] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. 2020. 6

[6] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 3

[7] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 3

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 1

[9] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 2

[10] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International*

Conference on Machine Learning, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 7 2021. 1

[11] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 516–533, Cham, 2022. Springer Nature Switzerland. 1, 2

[12] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12889–12899, Los Alamitos, CA, USA, 6 2021. IEEE Computer Society. 6

[13] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 1

[14] Xuwei Xu, Sen Wang, Yudong Chen, Yanping Zheng, Zhewei Wei, and Jiajun Liu. Gtp-vit: Efficient vision transformers via graph-based token propagation. 2023. WACV2024 oral. 6

[15] Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10818, 6 2022. 6

[16] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. 2

[17] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 3

E. Raw Data

A comprehensive overview of all training runs, encompassing all aspects such as model details, pretraining and finetuning resolutions, theoretical and empirical efficiency metrics, and accuracy measures, is provided in Table 6. This extensive table also includes a breakdown of non-default hyperparameters employed for both pretraining and finetuning stages. This data forms the foundation of our analysis and enables a deeper exploration of the nuances in model behavior and performance.

Table 6. Results and metrics as well as non-default hyperparameters of *all* of our ImageNet training runs.

Model	Taxonomy Class	Pretrain Res. [px]	Res. [px]	Params [Mil]	GFlops	Train Mem. [GB]	Inf. Mem. [GB]	Finetune Time [h*GPUs]	Throughput [ims/s]	Top-1 Acc. [%]	Top-5 Acc. [%]	Further Run Parameters
ViT-S/16	Baseline	192	224	22.06	4.66	107.27	1.45	19.3	3569.86(@2048)	82.22	96.54	
ViT-Ti/16	Baseline	192	224	5.72	1.28	54.21	0.50	12.9	7731.79(@4681)	73.78	92.85	
ViT-B/16	Baseline	192	224	86.59	17.68	221.36	2.20	30.0	1601.28(@2048)	84.66	97.38	fine-tune: [world_size=8]
ViT-S/16	Baseline	224	224	22.06	4.66	107.27	1.45	20.0	3582.39(@2048)	82.54	96.58	
ViT-B/16	Baseline	224	224	86.59	17.68	215.53	2.34	31.6	1614.04(@2048)	84.80	97.38	
ViT-Ti/16	Baseline	224	224	5.72	1.28	54.21	0.50	21.8	8312.30(@8192)	74.27	92.84	
ViT-Ti/16	Baseline	224	384	5.79	4.82	254.92	1.16	43.6	1903.29(@1024)	77.62	94.67	fine-tune: [world_size=8]
ViT-S/16	Baseline	224	384	22.21	15.76	505.88	4.73	69.5	829.86(@512)	84.25	97.36	fine-tune: [world_size=8]
DeiT-S/16	Baseline	192	224	22.06	4.66	107.43	1.45	21.9	3686.66(@2730)	82.03	96.40	
DeiT-S/16	Baseline	224	224	22.06	4.66	107.43	1.45	21.3	3713.59(@2048)	82.29	96.46	
DeiT-S/16	Baseline	224	384	22.21	15.74	505.89	2.11	61.9	996.94(@512)	84.01	97.29	fine-tune: [world_size=8]
ResNet50	Baseline (ResNet)	224	224	25.56	4.12	88.57	1.39	24.0	3135.04(@1360)	77.90	94.60	pre-train: [world_size=2]
Nystrom ViT-32-S/16	Low-Rank Attention	192	224	22.05	4.49	94.73	1.45	22.9	3611.83(@2048)	80.27	95.60	
Nystrom ViT-64-S/16	Low-Rank Attention	192	224	22.05	5.07	146.55	0.92	39.2	2754.91(@2048)	80.83	95.91	
Nystrom ViT-64-S/16	Low-Rank Attention	224	224	22.05	5.07	146.55	0.92	28.8	2749.68(@2340)	80.75	95.74	
Nystrom ViT-32-S/16	Low-Rank Attention	224	224	22.05	4.48	94.73	0.92	22.4	3707.31(@2048)	80.54	95.65	
Nystrom ViT-64-S/16	Low-Rank Attention	224	384	22.20	13.91	322.65	1.48	56.5	1320.54(@1024)	82.15	96.36	fine-tune: [world_size=8]
Nystrom ViT-32-S/16	Low-Rank Attention	224	384	22.20	13.01	247.53	1.48	50.1	1494.90(@1024)	81.99	96.37	fine-tune: [world_size=8]
Linformer-S/16	Low-Rank Attention	192	224	23.26	5.24	132.92	1.54	28.7	3399.36(@1365)	81.46	96.11	
Linformer-S/16	Low-Rank Attention	224	224	23.26	5.24	132.92	1.54	19.5	3394.80(@2048)	81.23	96.10	
XCiT-S-12/16	Low-Rank Attention	192	224	26.25	4.93	131.87	1.14	29.6	2173.73(@2048)	83.61	96.98	
XCiT-S-12/16	Low-Rank Attention	224	224	26.25	4.94		1.61	51.9	2037.84(@4096)	83.65	97.01	
XCiT-S-12/16	Low-Rank Attention	224	384	26.25	14.48	377.35	1.97	86.1	728.91(@1024)	84.77	97.50	fine-tune: [world_size=8]
SwinV2-Ti-W7/4	Sparse Attention	224	224	28.35	4.60	161.45	1.83	29.9	1758.66(@1024)	83.11	96.60	
SwinV2-Ti-W7/4	Sparse Attention	224	224	28.35	4.60	161.44	1.83	35.5	1775.66(@1024)	83.09	96.58	
Swin-Ti-W7/4	Sparse Attention	224	224	28.29	4.58	137.03	2.33	27.6	1921.73(@1024)	82.91	96.61	
Swin-S-W7/4	Sparse Attention	224	224	49.61	8.88	220.55	2.41	38.6	1191.97(@585)	84.87	97.43	both: [world_size=8]
Sinkhorn Cait-S-Bmax-32/16	Sparse Attention	192	224	25.61	4.50	101.27	1.07	18.6	3174.18(@1024)	81.01	96.11	
Sinkhorn Cait-S-Bmax-32/16	Sparse Attention	224	224	25.61	4.50	101.27	1.07	20.1	3168.82(@1638)	81.27	96.15	
Sinkhorn Cait-S-Bmax-32/16	Sparse Attention	224	384	25.76	13.21	291.48	1.75	52.2	1249.14(@1024)	81.77	96.43	fine-tune: [world_size=8]
Routing ViT-S/16	Sparse Attention	192	224	22.04	4.34	132.53	1.02	23.7	2398.10(@1365)	73.59	92.38	
Routing ViT-S/16	Sparse Attention	224	224	22.04	4.35	134.82	1.03	22.7	2341.10(@2048)	73.92	92.68	
Routing ViT-S/16	Sparse Attention	224	384	22.19	12.75	375.20	1.78	66.5		72.55	91.95	fine-tune: [world_size=8]
WaveViT-S	Sparse Attention	192	224	22.24	4.73	240.14	1.48	57.9	767.14(@1024)	83.91	97.10	fine-tune: [world_size=8]
WaveViT-S	Sparse Attention	224	224	22.24	4.73	238.28	1.62	55.6	767.12(@1024)	83.61	96.92	
Informer-S/16	Sparse Attention	224	224	22.05	4.37	123.21	1.04	42.6	1510.20(@1024)	73.47	92.38	both: [amp=False]
Synthesizer FD-S/16	Fixed Attention	192	224	19.32	3.94	129.84	0.90	35.8	710.57(@2048)	75.45	93.17	
Synthesizer FD-S/16	Fixed Attention	224	224	19.32	3.94	129.84	0.90	35.9	711.33(@2048)	75.51	93.04	
Synthesizer FR-S/16	Fixed Attention	192	224	18.74	3.80	68.19	0.86	16.3	6065.16(@2048)	78.80	95.15	
Synthesizer FR-S/16	Fixed Attention	224	224	18.74	3.80	68.19	0.86	15.7	6029.23(@2048)	79.07	95.14	
Synthesizer FR-S/16	Fixed Attention	224	384	19.33	12.35	199.19	1.42	57.5	2018.33(@1024)	79.77	95.39	fine-tune: [world_size=8]
PolySA-S/16	Kernel Attention	192	224	22.09	4.29	82.28	0.92	15.3	5404.60(@2336)	78.44	94.50	
PolySA-S/16	Kernel Attention	224	224	22.09	4.29	82.28	0.92	14.9	5358.89(@2048)	78.34	94.50	
PolySA-S/16	Kernel Attention	224	384	22.29	12.56	239.11	1.48	42.9	1836.55(@512)	79.74	95.27	fine-tune: [world_size=8]
SLAB-S/16	Kernel Attention	224	224	23.00	4.40	109.65	0.60	30.1	2569.86(@1024)	78.70	95.09	
Hydra-S/16	Kernel Attention	224	224	22.06	4.29	103.11	0.58	22.5	6357.89(@512)	77.84	94.26	
EfficientFormerV2-S0	Hybrid Attention	192	224	3.42	0.44	60.09	0.69	28.9	954.46(@4096)	71.45	91.10	
EfficientFormerV2-S0	Hybrid Attention	224	224	3.42	0.44	60.09	0.69	28.1	993.24(@4096)	71.53	91.19	
EfficientFormerV2-S0	Hybrid Attention	224	382	3.43	1.42	189.32	1.47	57.1	726.02(@1024)	74.43	92.92	fine-tune: [world_size=8]
CvT-13	Hybrid Attention	192	224	20.00	4.65	159.31	2.11	26.1	2208.64(@512)	81.53	96.27	fine-tune: [lr=0.0002]
CvT-13	Hybrid Attention	224	224	20.00	4.65	161.11	1.96	27.4	2218.90(@512)	82.35	96.47	both: [world_size=8]
CvT-13	Hybrid Attention	224	224	20.00	4.65	161.11	1.96	27.2	2218.02(@512)	82.30	96.41	both: [world_size=8]
CoaT-Ti	Hybrid Attention	192	224	5.50	4.45	234.17	1.61	58.0	926.38(@512)	78.10	94.66	fine-tune: [world_size=8]
CoaT-Ti	Hybrid Attention	224	224	5.50	4.45	234.17	1.61	52.9	989.03(@512)	78.42	94.75	both: [world_size=8]
Efficient ViT-B-2	Hybrid Attention	224	224	24.33	1.60	145.25	0.70	29.1	4182.78(@1024)	81.52	96.14	
NextViT-S	Hybrid Attention	224	224	31.76	5.82	111.97	0.69	35.4	3440.40(@2048)	83.92	97.18	
ResT-S	Hybrid Attention	224	224	13.68	2.15	143.49	0.63	28.5	4420.31(@512)	79.92	95.50	both: [amp=False]
Mixer-S/16	Non-Attention Shuffling	224	224	18.53	3.82	56.98	0.92	39.0	6197.41(@1024)	76.20	93.50	pre-train: [lr=0.002, opt_eps=1e-06, world_size=8]
FocalNet-Ti-Srf	Non-Attention Shuffling	192	224	28.43	4.51	137.65	3.47	24.9	1887.22(@512)	83.21	96.84	
FocalNet-S-Srf	Non-Attention Shuffling	192	224	49.89	8.77	222.67	5.04	36.1	1279.22(@512)	84.91	97.42	fine-tune: [world_size=8]
FocalNet-S-Srf	Non-Attention Shuffling	224	224	49.89	8.77	222.67	5.04	38.9	1171.29(@512)	84.91	97.38	fine-tune: [world_size=8]
FocalNet-Ti-Srf	Non-Attention Shuffling	224	224	28.43	4.51	139.96	3.32	29.9	1884.82(@512)	83.23	96.74	fine-tune: [world_size=8]
FocalNet-Ti-Srf	Non-Attention Shuffling	224	224	28.43	4.51	139.96	3.32	25.9	1886.80(@512)	82.11	96.15	fine-tune: [world_size=8, lr=0.003]
FocalNet-Ti-Srf	Non-Attention Shuffling	224	382	28.43	12.22	381.64	8.36	70.5	700.12(@256)	84.03	97.17	fine-tune: [world_size=8]
FocalNet-Ti-Srf	Non-Attention Shuffling	224	382	28.43	12.22	381.64	8.36	76.0	701.76(@512)	83.00	96.56	fine-tune: [world_size=8, lr=0.003]
SwiftFormer	Non-Attention Shuffling	224	224	5.87	1.03	69.87	0.62	22.8	6239.36(@2048)	76.41	93.84	
SwiftFormer	Non-Attention Shuffling	224	384	5.87	3.02	204.27	1.66	54.5	2145.56(@512)	78.35	94.77	
FastViT-12	Non-Attention Shuffling	224	224	9.47	1.45	94.82	0.71	29.3	3629.08(@2048)	78.77	94.83	
FastViT-12	Non-Attention Shuffling	224	384	9.47	4.25	277.11	2.00	88.6	1260.02(@512)	80.32	95.49	
Efficient Mod-S	Non-Attention Shuffling	224	224	8.99	1.11	101.31	0.62	26.6	3726.89(@8192)	80.21	95.63	
FNet-S/16	Fourier Attention	192	224	14.96	2.96	67.43	1.42	21.4	4723.61(@2048)	73.00	91.54	
FNet-S/16	Fourier Attention	224	224	14.96	2.96	67.43	1.42	20.1	4714.13(@2048)	73.12	91.68	
FNet-S/16	Fourier Attention	224	384	15.11	8.70	195.85	1.34	42.0	1775.41(@1024)	74.99	92.78	fine-tune: [world_size=8]
GFNet-S/16	Fourier Attention	192	224	24.87	4.57	100.30	1.42	24.5	3335.00(@4096)	80.70	95.51	
GFNet-S/16	Fourier Attention	192	224	24.87	4.57	100.30	1.42	23.2	3210.40(@2048)	80.69	95.45	
GFNet-S/16	Fourier Attention	224	224	24.87	4.57	100.30	1.42	23.5	3173.66(@3264)	79.69	94.77	fine-tune: [lr=0.003]
GFNet-S/16	Fourier Attention	224	224	24.87	4.57	100.30	0.94	18.0	3433.93(@4096)	81.33	95.75	
GFNet-S/16	Fourier Attention	224	384	27.93	13.46	291.09	1.47	56.0	1155.27(@1024)	83.12	96.68	fine-tune: [world_size=8]
GFNet-S/16	Fourier Attention	224	384	27.93	13.46	291.09	1.47	56.0	1118.63(@512)	81.25	95.56	fine-tune: [world_size=8, lr=0.003]
AFNO-S/16	Fourier Attention	224	224	15.87	3.11	106.97	1.43	24.0	3549.92(@1024)	78.55	94.86	
AFNO-Ti/16	Fourier Attention	224	224	4.17	0.81	55.11	1.27	19.8	6644.17(@8192)	68.87	89.86	both: [world_size=8]
AFNO-Ti/16	Fourier Attention	224	384	4.25	2.39	155.91	3.68	51.8	2295.85(@2048)	72.37	91.88	both: [world_size=8]
AFNO-S/16	Fourier Attention	224	384	16.01	9.13	307.42	4.02	69.2	1232.72(@1024)	80.84	95.96	fine-tune: [world_size=8]
EViT-S/16	Token Removal	224	224	22.05	4.66	100.18	1.45	31.1	3757.28(@4096)	82.29	96.48	
EViT-S/16	Token Removal	224	224	22.05	4.66	100.18	0.92	25.0	3830.13(@2048)	81.98	96.41	
EViT-S/16	Token Removal	224	384	22.20	15.75	487.29	2.06	67.3	938.44(@512)	84.18	97.26	fine-tune: [world_size=8]
Dynamic ViT-S/16	Token Removal	192	224	22.77	4.09	132.53	0.92	25.0	4250.67(@2048)	80.95	95.88	pre-train: [lr=0.001]
Dynamic ViT-S/16	Token Removal	224	224	22.77	4.09	132.53	0.92	22.2	4238.73(@2340)	81.09	95.81	pre-train: [lr=0.001, opt_eps=1e-06]
EViT Fuse-S/16	Token Merging	192	224	22.03	4.66	100.18	0.92	23.3	4013.41(@2048)	82.20	96.52	
EViT Fuse-S/16	Token Merging	224	224	22.05	4.66	100.18	1.45	17.6	3558.60(@2048)	81.96	96.31	
EViT Fuse-S/16	Token Merging	224	384	22.20	15.75	487.29	2.06	61.9	942.82(@512)	83.90	97.21	fine-tune: [world_size=8]
ToMe-S-R-8/16	Token Merging	192	224	22.06	3.47	85.26	0.93	19.1	4216.95(@2048)	82.01	96.37	
ToMe-S-R-8/16	Token Merging	224	224	22.06	3.47	85.26	0.93	18.3	4214.67(@2048)	82.11	96.37	
ToMe-S-R-8/16	Token Merging	224	384	22.21	14.32	474.04	2.08	65.9	865.81(@512)	84.15	97.13	fine-tune: [world_size=8]
Token Learner-8-50-S/16	Summary Tokens	192	224	12.01	2.58	58.57	0.73	15.5	6925.34(@4096)	76.85	93.89	
Token Learner-8-75-S/16	Summary Tokens	192	224	17.34	3.77	85.11	0.83	16.2	4553.70(@2048)	80.69	95.58	
Token Learner-8-75-S/16	Summary Tokens	224	224	17.34	3.77	85.11	0.83	19.9	4543.89(@2048)	80.66	95.59	
Token Learner-8-50-S/16	Summary Tokens	224	224	12.01	2.58	58.57	0.73	12.1	6975.32(@4096)	77.35	94.27	
Token Learner-8-50-S/16	Summary Tokens	224	384	12.16	8.42	265.98	1.92	44.5	1581.53(@512)	80.07	95.46	fine-tune: [world_size=8]
Token Learner-8-75-S/16	Summary Tokens	224	384	17.48	12.36	386.15	2.02	61.2	1168.62(@512)	82.67	96.40	fine-tune: [world_size=8]
STViT-Swin-Ti-W-7/4	Summary Tokens	224	224	28.30	3.54	130.17	1.91	25.0	2039.02(@512)	82.22	96.21	
CaiT-S-24	Summary Tokens	192	224	46.92	9.45	292.59	1.38	42.2	1053.25(@2048)	84.45	97.35	fine-tune: [world_size=8]
CaiT-S-24	Summary Tokens	224	224	46.92	9.45	292.59	1.38	42.1	1055.92(@2048)	84.91	97.44	both: [world_size=8]
CaiT-S-24	Summary Tokens	224	224	46.92	9.45	292.59	1.38	42.0	1055.97(@2048)	84.76	97.43	both: [world_size=8]
Switch ViT-8-S/16	More MLPs	192	224	121.35	4.67	128.64	2.81	24.2	3611.77(@4096)	82.19	96.23	
Switch ViT-8-S/16	More MLPs	224	224	121.35	4.67	128.64	2.81	24.0	3351.59(@4096)	82.39	96.19	pre-train: [lr=0.002]
Switch ViT-8-S/16	More MLPs	224	224	121.35	4.67	137.69	2.66	25.9	3575.28(@4096)	82.30	96.23	pre-train: [lr=0.002] fine-tune: [world_size=8]
Switch ViT-8-S/16	More MLPs	224	384	121.49	15.78	558.24	4.01	78.8	862.82(@1024)	84.23	97.07	pre-train: [lr=0.002] fine-tune: [world_size=8]
Hi ViT-Ti/16	More MLPs	224	224	4.94	1.26	70.75	0.49	19.4	6720.79(@8192)	76.13	93.90	
Hi ViT-Ti/16	More MLPs	224	224	4.94	1.26	70.75	0.49	21.7	6714.25(@4096)	75.94	93.76	