IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 December 2024, accepted 21 January 2025, date of publication 29 January 2025, date of current version 11 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3536471

RESEARCH ARTICLE

Patch and Model Size Characterization for On-Device Efficient-ViTs on Small Datasets Using 12 Quantitative Metrics

JURN-GYU PARK[1], (Associate Member, IEEE), AIDAR AMANGELDI[1], NAIL FAKHRUTDINOV[1], MERUYERT KARZHAUBAYEVA[2], AND DIMITRIOS ZORBAS[1], (Member, IEEE)

[1]School of Engineering and Digital Sciences, Nazarbayev University, 010000 Astana, Kazakhstan
[2]Heinz College of Information Systems and Public Policy, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Corresponding author: Jurn-Gyu Park (jurn.park@nu.edu.kz)

This work has emanated from research conducted with the financial support of the Ministry of Science and Higher Education of the Republic of Kazakhstan, AP23487072 (“Leveraging IoT Mesh Networks for Machine Learning Knowledge Transfer.”) and of Nazarbayev University (NU), under grant 021220FD0851 (FDCRGP).

ABSTRACT Vision transformers (ViTs) have emerged as a successful alternative to convolutional neural networks (CNNs) in deep learning (DL) applications for computer vision (CV), particularly excelling in accuracy on large-scale datasets within high-performance computing (HPC) or cloud domains. However, in the context of resource-constrained mobile and edge AI devices, there is a lack of systematic and comprehensive investigations into the challenging optimizations for both device-agnostic (e.g., accuracy and model size) and device-related (e.g., latency, memory usage, and power/energy consumption) multi-objectives. To resolve this problem, we first 1) introduce five device-agnostic (DA) and seven device-related (DR) quantitative metrics, 2) using which we thoroughly characterize the effects of ViT hyper-parameters on small datasets in terms of patch size and model size, and then 3) propose a simple yet effective optimization technique called the hierarchical and local (HelLo) tuning method for efficient ViTs. The results show that our method achieves significant improvements of up to 85% in MACs, 67.2% in inference latency, 77.7% in train latency/time, 63.3% in GPU memory, 73.8% in energy consumption, and 263.0% in FoM, with minimal accuracy degradation (up to 2%).

INDEX TERMS Deep learning, efficient vision transformers (ViTs), edge-AI, on-device ML, multi-objective optimization, characterization, mobile devices, embedded systems.

I. INTRODUCTION

Vision transformers (ViTs) have gained significant attention in the field of computer vision due to their ability to effectively leverage self-attention mechanisms, which allow them to capture global dependencies in an image with improved interpretability [1].

Unlike traditional convolutional neural networks (CNNs) that process images through localized filters, ViTs treat the image as a sequence of patches and learn relationships between these patches, enabling a more flexible and comprehensive understanding of the visual data. This fundamental shift in the approach has led to notable improvements in performance across a wide range of ViT variants [1], [2], [3], [4], [5] using large-scale datasets such as ImageNets [6], [7] and JFTs [5], [8], and other CV tasks of object detection [9], image segmentation [10], and video analysis [11], establishing ViTs as a powerful alternative against conventional CNN architectures.

However, designing efficient-ViTs for mobile and edge devices is more challenging not only due to the high quadratic computation and memory requirements of ViT [1], but also

The associate editor coordinating the review of this manuscript and approving it for publication was Ugur Guvenc.

because of the need to pursue multiple objectives of accuracy, latency, memory usage, and energy consumption, etc. Additionally, there is a growing need to train efficient deep learning (DL) models on embedded and edge devices [59], [60]. This is especially crucial and indispensable in transfer learning (TL) based fine-tuning training [61] and federated learning (FL) modeling [62] in privacy-sensitive domains such as healthcare and medicine, where sharing datasets in the cloud is not an option. Moreover, the high computational demands of ViTs have raised concerns about their feasibility in mobile and edge computing scenarios, where traditional CNNs still have advantages over standard ViT architecture due to their accuracy-efficiency trade-offs [12].

Unlike HPC and cloud domains, mobile and edge (on-device) AI systems may have the following practical constraints:

- ***Training ViT models on Small Datasets without Knowledge Distillation (KD)***: In domains like science and medicine, large datasets comparable to the size of ImageNet [7] are rarely available, making it difficult to train teacher models for KD.
- ***Limitation of Training on Resource-constrained Edge (On-Device) Platforms***: In medical domains, privacy concerns often prevent sharing datasets in the cloud for training. However, it is very challenging to collect data and train models on resource-constrained edge devices (i.e., on-device training).
- ***Design of Efficient-ViTs using Multiple quantitative Metrics***: Resource-constrained mobile and edge AI platforms must meet multiple objective metrics, including not only accuracy but also latency (inference and training time), memory usage, energy consumption, and other combinations of these objectives.

Given the memory and energy constraints of mobile and edge AI devices, we set the goal to find energy- and memory-efficient ViT setups that maintain comparable accuracy with improved inference and training latencies to baseline ViTs without significant accuracy degradation. To achieve this, we propose a hierarchical and local optimization method (HelLo), characterizing ViTs on small datasets based on patch size and model size hyper-parameters to efficiently address multiple objectives, including accuracy, memory usage, and energy consumption. Therefore, our paper makes the following contributions:

- Employ twelve quantitative metrics, which includes five device-agnostic (DA) and seven device-related (DR) metrics for multi-objective.
- Using the metrics, analyze and characterize the effect of the patch size (P) and the model size of the number of layers (L), the size of the input dimension (D), the number of attention heads (H), and the size of the multi-layer perceptron (M).
- Propose a hierarchical and local ViT model optimization method (HelLo) and conduct an ablation study, locally optimizing the multi-objective quantitative metrics.
- Our results show improvements of up to 85% in MACs and 263.0% in FoM, with minimal accuracy loss. The code for our study is available here.[1]

The rest of the paper is organized as follows: Section II presents our motivation and summarizes related work. In Section III, we introduce the experimental setup, methodology, and quantitative metrics. Section IV summarizes our preliminary results, and Section V presents the results, analysis and key findings. In Section VI, we present discussion with future work. Finally, Section VII concludes the paper.

II. MOTIVATION AND RELATED WORK

A. BACKGROUND

As shown in Figure 1, the architecture of ViTs consists of three main components: patch embedding, transformer encoder, and classification head. **Patch Embedding**: The input images of size $H \times W \times C$ (H: height, W: weight, C: channel) are divided into N patches of patch size $P \times P$. Each patch is flattened and mapped to a D-dimensional vector using a learnable linear projection. Positional embeddings are added to retain spatial information. **Transformer Encoder**: The embedded patches are fed into a stack of transformer layers, L. Each layer comprises multi-head self-attention (MHSA) modules and feed-forward neural networks (FFN), both equipped with layer normalization and residual connections. **Classification Head**: A special `[CLS]` token is prepended to the patch embeddings, and its final representation is used for classification tasks.

B. MOTIVATION

Considering the three practical constraints in mobile and edge AI scenarios, our work focuses on optimizing the ViT-Tiny model rather than larger models like ViT-S, ViT-B, and ViT-L (Table 1), as it is more relevant to resource-constrained mobile and edge devices. Based on the original ViT [1] and its variant (DeiT) [2], Equation 1 presents variants of ViT models as the function of the input **patch size** (P) and the **model size** of the number of layers (L), the size of the input dimension (D), the number of attention heads (H), the size of the multi-layer perceptron (M).

$$\text{Model Variants} = f(P) + f(L, D, H, M) \tag{1}$$

Considering resource-constrained mobile and edge (on-device) domains, we try to optimize multiple objectives for efficient ViT models with negligible accuracy degradation, using ViT-Tiny (Table 1) to address the following two motivating questions:

- Given any baseline ViT model, *how can we quickly optimize a locally optimal multi-objective efficient-ViT model without or within negligible k% accuracy degradation (e.g., 2%) using quantitative metrics for multiple objectives?*

[1] https://github.com/esainulab/efficientvit

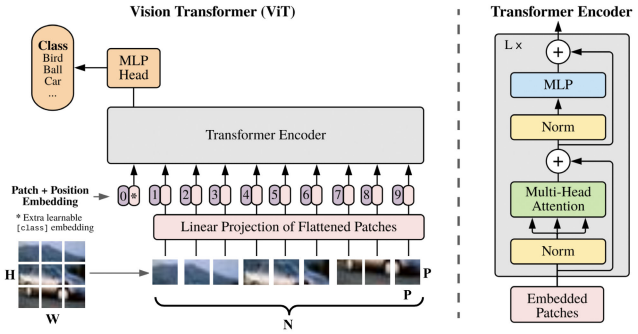


FIGURE 1. ViT architecture (Deployed from [1] and modified).

TABLE 1. ViT model variants [1], [2].

Model	Patch Size	Layers Size	D Size	Heads Size	MLP Size	Params
ViT-Tiny	**16/32**	**12**	**192**	**3**	**768**	**6M**
ViT-S	16/32	12	384	6	1536	22M
ViT-B	16/32	12	768	12	3072	86M
ViT-L	16/32	24	1024	16	4096	307M

- *Which hyper-parameter(s) from the patch size (P) and the model size (L, D, H, or M) are most significant from the perspective of improvements based on the quantitative metrics?*

To answer these two motivating questions, we characterize and optimize the effects of the hyper-parameters, and provide key findings and optimization guidelines.

C. RELATED WORK

Efficient ViTs for Mobile or Edge devices: Although the original large-scale datasets based ViT [1] and its variants [2], [3], [4], [5], [13] do not outperform efficient CNNs like MobileNets v2/v3 [14], [15] in an empirical study [16], more recent efficient-ViTs for mobile or edge devices such as EdgeNeXt [17], EdgeVits [12], EfficientFormer v1/v2 [18], [19], and EfficientVit [20] compete with the speed of CNN-based MobileNet v2 [14] in an efficiency evaluation study on a mobile platform [21]. However, multi-objective metrics-based systematic investigations of efficient-ViTs using latency, memory usage, and energy consumption metrics on mobile or edge GPU devices are very rare, while device-agnostic evaluations such as accuracy, the number of parameters and MACs are relatively well studied, independent of device type. Among the many efficient-ViT studies, only EdgeViT [12] evaluated latency and energy on a mobile Android phone (CPUs on Samsung Galaxy S21 with a Snapdragon 888 chip) using both inference latency and energy consumption (not on GPU but on CPUs), while [17] evaluated only latency on an edge GPU device (TensorRT engines on Nvidia Jetson Nano), [18], [19], [21] evaluated only latency on mobile phones (NPU/CPUs on iPhone 12 with a 14 bionic chip/Pixel 6 Android/Google Pixel 4 Android), and [20] evaluated the device-related inference latency on two edge GPUs (Nvidia Jetson Nano and AGX Orin using TensorRT).

Training Efficient-ViTs on Small Datasets w.o. KD: As an another direction of dataset-constrained and privacy-issued domains such as science and medical domains, training the efficient-ViTs on small datasets (e.g., CIFAR-10 and CIFAR-100) is indispensable. The line of work mainly focuses on data augmentation [22], self-supervised techniques [23], [24], and more-efficient self-attention design concepts [25], [26], [27]. Despite the challenging constraints of the domains, related works on model evaluations using latency, memory usage, and energy consumption metrics on resource-constrained mobile or edge GPU devices are very rare.

Multi-objective Optimization for Efficient-ViTs: Although there are some works for CNN neural architecture search (NAS) using multi-objectives such as DPP-Net [28] and MONAS [29], it is very rare in efficient-ViT studies. The empirical study [16] employed the metrics of accuracy, inference latency, memory footprint, and energy consumption to compare efficient-CNNs and lightweight ViT variants, and provided insightful findings for efficient-ViT optimization.

Local and Hierarchical Search: The most basic form of greedy local search is often called the hill-climbing algorithm [30]. We adopt straightforward concepts of hierarchical and local search for our optimization method. Recently, the local search was used in NAS [31], [32] as a strong baseline, compared to the random search. Moreover, with the computational benefits of hierarchical search by examining an idealized case [30], GLiT [33] proposed a hierarchical NAS, tackling the problem caused by huge search space.

Note that our work is orthogonal in terms of goal and method. Our goal is not to design a new efficient-ViT backbone, but to optimize memory- and energy-efficient ViTs with improved inference and training latencies plus negligible accuracy drop in any base ViT model using multiple objective (including multi-objective) metrics for resource-constrained mobile or edge GPU devices. Moreover, our method is simple yet effective using a hierarchical and local optimization concept (an approach of fast and lightweight tuning with characterizations rather than a black-box model-based NAS approach).

III. METHODOLOGY

This section has four subsections: **Experimental setup** is described firstly, because it establishes the foundation of the study by detailing the hardware platforms, software frameworks, and utilized datasets. This is essential for understanding the methodology context in which the experiments are conducted. **Optimization space** is introduced to define the specific hyper-parameters that are being optimized, providing clarity on the scope of the study. **Optimization strategy** is outlined to describe the specific techniques applied to search the hierarchically and locally optimal hyper-parameters within the defined space. **Quantitative metrics** are discussed in detail to explain the measurement and evaluation of the results, as well as the assessment

TABLE 2. Specifications of experimental platforms.

Mobile Laptop	
GPU	3840-core NVIDIA Ampere™ GPU architecture with 30 SMs (128-core per SM), 1.7 Ghz
CPU	Intel Quad Core™ i5-11 Gen., 3.1 GHz
Memory	CPU: 16GB DDR4 SDRAM Memory, 1.6 GHz GPU: 6GB GDDR6 Memory, 1.6 GHz
Storage	512GB SSD
Edge AI Device	
GPU	256-core NVIDIA Pascal™ GPU architecture with 2 SMs (128-core per SM), 1.3 Ghz
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore, 2.04 Ghz
Memory	8GB 128-bit LPDDR4, Memory 1.9 GHz - 59.7 GB/s
Storage	32GB eMMC 5.1
Power	7.5W / 15W

of the improvements of the optimization efforts. This structure ensures systematic methodology phases from the experimental setup to quantitative and fair evaluation of the results.

A. EXPERIMENTAL SETUP

Experimental Platforms: As shown in Table 2, we utilize two platforms — NVIDIA RTX3060 6GB based laptop and the NVIDIA Jetson TX2 edge device — to measure the performance of ViT-Tiny models. The laptop platform measures all device-agnostic (DA) metrics such as accuracy, number of parameters, and number of MACs; and it also measures all device-related (DR) metrics except for those metrics that are directly related to power consumption.

The edge device measures only device-related (DR) metrics, including power combined metrics. We conduct a separate setup for experiments on NVIDIA Jetson TX2, utilizing *tegrastats* tool [34] that provides detailed power consumption and memory logs at 100 ms intervals, along with real-time data on CPU and GPU utilization and clock frequencies.

Regarding the experimental evaluations, the default (baseline) setup was run five times, and we took the trimmed average by removing the minimum and maximum values, whereas for all other setups, we conducted only a single run.

Software Frameworks: ViT models are implemented and tested primarily on the TensorFlow framework [35] using the GitHub codes and additionally on the PyTorch framework [36] using the codes [25].

Dataset and Applications: CIFAR-10 image classification dataset [37] is used as a small dataset for this work. It consists of 60,000 color images with 10 different classes, with each class representing a distinct object or category. Note that the original images have a resolution of 32 by 32 pixels, but they are resized to 224 by 224 pixels, following the common practice [1], [38] and escaping patch size confusions compared to the ImageNet.

B. OPTIMIZATION SPACE

Our optimization space is based on a hierarchical manner of **high-level** input patch size (P) and **low-level** architecture search of each model size, which includes the number of layers (L), the hidden dimension size (D), the number of heads (H), and the MLP size (M). As shown in Table 3, the input patch sizes of 16 and 32, along with the model sizes to be searched, are described with the underlined default configurations of ViT-tiny, from the perspective of the transformer (ATT + FFN), attention (ATT), and FFN layers.

TABLE 3. Optimization space.

High-level: Input Patch size		
Input	Patch size	{16, 32, 56, 112} → {16, 32}
Low-level: Model size		
Transformer	#Layer	{2, 4, 6, 8, 10, <u>12</u>}
Attention	D size	{96, 128, <u>192</u>, 256}
Attention	#Heads	{1, 2, <u>3</u>, 4, 5, 6}
FFN	MLP size	{192, 384, <u>768</u>, 960, 1152}

Based on the design goal of fast optimization for efficient ViTs with negligible or minimal accuracy degradation plus improved inference and training latencies, the design space in Table 3 is considered. The patch sizes of 16 and 32 are selected in a hierarchical manner, based on the preliminary results of the patch sizes of 16, 32, 56, and 112, excluding 56 and 112 due to significant accuracy degradations. We also considered smaller model sizes than the default underlined configurations. However, for the attention (ATT) or FFN layer, we also consider bigger model sizes (i.e., D: 256, H: 4, 5 and 6, M: 960 and 1152) than the default underlined configurations, but with smaller (or equal only in H) model sizes than ViT-Small, considering broader search spaces (e.g., a reduced L but an increased D/H/M).

C. OPTIMIZATION STRATEGY: HIERARCHICAL AND LOCAL SEARCH WITH CHARACTERIZATION (HELLO)

In order to quickly search local-optimal efficient ViTs within k% (e.g. 2%) accuracy degradation using multiple objective goals, we propose a hierarchical- and local-search based optimization technique, as shown in Figure 2: 1) firstly, we investigate the input patch size effects of ViT-Ti/16 and ViT-Ti/32, as a high-level search. Then, 2) we analyze and characterize the low-level search for the effects of L, D, H, and M. 3) Lastly, using the model sizes of local-optima, we conduct an ablation study on ViT-Ti/16 and ViT-Ti/32.

1) HIGH-LEVEL: INPUT PATCH SIZE

The size of input patches affects the accuracy/speed trade-offs, generally leading to higher accuracy with smaller patches at a greater computational cost [39]. However, strictly speaking, a local optimum patch size is dependent on the input image resolution [25] and image characteristics (e.g., simple or complex images) [38]. Considering the practical effects of the input patch size, we conduct a comprehensive comparison of ViT-Ti/16 and ViT-Ti/32 in terms of training epochs, frameworks (TensorFlow and PyTorch), and various patch size characteristics.

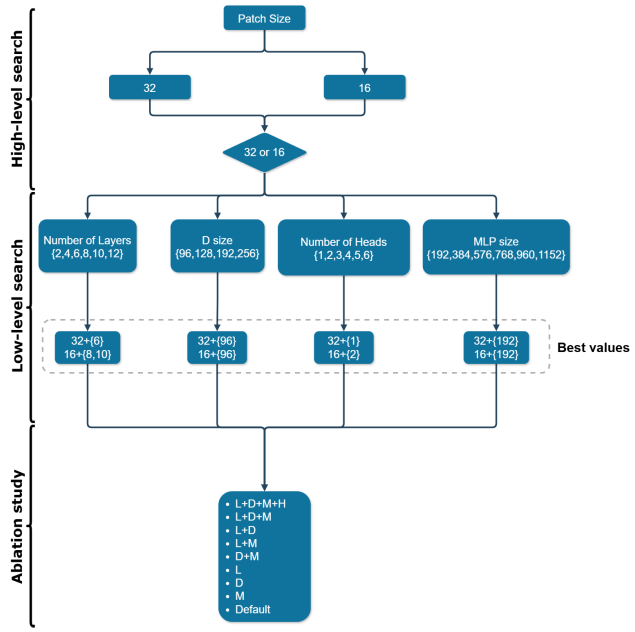


FIGURE 2. Optimization strategy.

2) LOW-LEVEL: MODEL SIZE

In the low-level search, firstly we investigate the layer (L) effects from the perspective of transformer (ATT + FFN). Then, the hidden dimension (D) size and the number of heads (H) in ATT layers are investigated. Lastly, we add the effects of the MLP size (M) in FFN layers. The optimization space is chosen manually based on the design goal of optimizing more efficient ViTs with minimal accuracy degradation plus improved inference and training latencies; and, we mostly select reduced model sizes but include some bigger sizes as well (e.g., in D, H and M), compared to the default.

3) ABLATION STUDY

Using the local optima found in each factor of L, D, H, and M, we conduct an ablation study to investigate the results of the combinations in Figure 2. Then, we find one best optimal and provide optimization guidelines for specific design goals according to the results of the locally optimal combinations.

D. QUANTITATIVE METRICS

For fair and comprehensive evaluations of device-agnostic (DA) and device-related (DR) performances, we employ twelve quantitative metrics for multiple objectives. As shown in Table 4, we specify four types of metric categories: device-agnostic single-objective (DA-Sin.), device-agnostic multiple-objective (DA-Mul.), device-related single-objective (DR-Sin.), and device-related multiple-objective (DR-Mul.) metrics.

1) DA-SIN

Accuracy, the number of parameters (*Params*), and the number of MACs (*MACs*) are adopted. While *Params* is directly provided by TensorFlow framework's output,

TABLE 4. Twelve quantitative metrics.

Type	Metric	Description
DA.-Sin	Accuracy (Acc.)	The accuracy after convergence of training on Laptop (100 epochs for ViT-Ti16 and 150 epochs for ViT-Ti32)
	Params	The number of parameters
	MACs	The number of Multiply-and-Add operations (Table 5)
DA-Mul.	Information Density (ID) [40]	$\frac{Accuracy}{Params}$ (2)
	NetScore [41]	$20 \log \left(\frac{Accuracy^2}{Params^{0.5} \cdot MACs^{0.5}} \right)$ (3)
DR-Sin.	Inf. latency (ms)	Inference time per batch-size (32 on Laptop and 16 on Edge; measured in milliseconds.)
	Tr. latency (hours)	Laptop: Total training time on 100 epochs for ViT-Ti/16 and 150 epochs for ViT-Ti/32 Edge: Total training time on 5 epochs (both patches)
	Memory-Usage	Laptop: GPU DRAM Edge: Average memory consumption of integrated DRAM - 1061(RAM_{base})
	Power (mW)	Average total power consumption (CPU+GPU+DRAM) throughout the training time.
DR-Mul.	Energy per epoch (J)	$Energy = Power \cdot Tr./ep.$ (4)
	Acc.-over-Latency (AoL) [42]	$AoL = \frac{Accuracy}{Inference}$ (5)
	Figure of Merit (FoM) [43]	$FoM = \frac{Accuracy}{Power \cdot Inference}$ (6)

TABLE 5. MACs calculation (***P***: patch size, ***N***: number of patches (i.e., input tokens), ***D***: hidden dimension size of tokens, ***M***: MLP size, ***L***: Number of layers). (Based on [44] and the GitHub code [45] of [46]).

Model	Sub-model	Equation
Embedding	Tokenizer	$3(N-1)DP^2$
Attention	Pre_L $Att(Q,K,V)$ Post_L	$\{ 3ND^2 +$ $2N^2D +$ $ND^2 \} \cdot L$
MLP	MLP	$2NDM \cdot L = 8ND^2 \cdot L$ (if $M = 4D$)
MLP Head	Class Head	$D \cdot \#Classes$ [2]

MACs is calculated manually following the method used in Table 5, where the embeddings creation, the attention block involvement with the projection of embeddings, calculation of attention scores, weighted averaging based on these scores, and projection of the result are processed. The MLP block and classification head consist of two linear layers. In the first layer of MLP, each embedding is projected from D size to a higher dimension (MLP size), and the second layer projects embeddings back from MLP size to D size.

2) DA-MuL

Information density (*ID*) [40] quantifies the trade-off between accuracy and *Params*, while *NetScore,bib:41* quantifies the trade-off between accuracy and model complexity, (i.e., *Params* and *MACs*). Another difference is that *NetScore* has different weights between accuracy and *Params*/*MACs* with the specific logarithmic format in Table 4.

[2]In our real calculation, $D \cdot M + M \cdot \#Classes$ is used due to one more added dense layer in our code.

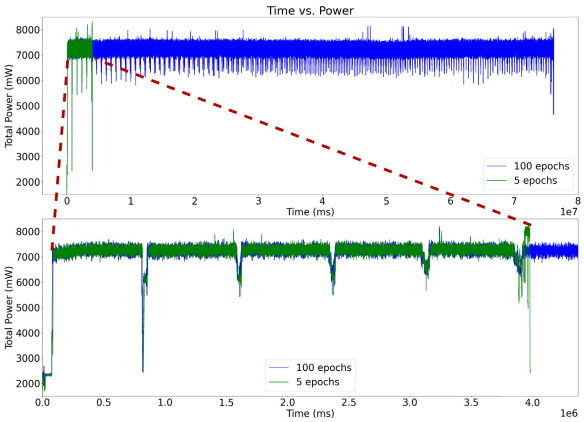


FIGURE 3. Comparison of power consumption between 100 (blue) and 5 (green) epochs. (Below is the zoomed 5 epochs.)

3) DR-SIN

Inference Latency is the inference time of each batch size (32 on Laptop and 16 on TX2). *Training Latency* is the total training time (hours) to complete the training until convergence. Therefore, it depends on the number of epochs (i.e., 100 epochs for ViT-Ti/16 and 150 epochs for ViT-Ti/32 in this study). *Memory-Usage* shows the amount of memory during the model training phase and is different for platforms - the laptop uses the discrete GPU memory, while the NVIDIA TX2 uses the integrated GPU memory (i.e., shared memory for both CPU and GPU). Therefore, *nvidia-smi,bib:47* is utilized to measure GPU memory consumption on the laptop, and the *tegrastats,bib:34* is utilized to measure the integrated memory (i.e., CPU+GPU memory) on the edge device.

Power Consumption: The measurement for power consumption is conducted on the TX2 edge device during the training phase using the total power of CPU, GPU, and DRAM power sensors. Due to time constraints on the edge device (which has a 3-5× longer training time per epoch compared to the laptop during the training phase), we reduce the number of epochs to 5, based on the observations in Figure 3. We observe that there is no difference between 5 and 100 epochs in terms of the average power. It remains almost constant regardless of the number of epochs (i.e., provided power curves of both cases are almost the same).

4) DR-MuL

Accuracy-over-Latency (AoL),bib:42 is a multi-objective metric that shows the trade-off between the *Accuracy* and the *Inference Latency*. *Figure of Merit (FoM)* [43] is a multi-objective metric that shows the trade-off between *Accuracy*, *Power*, and *Inference Latency*. *Energy Consumption* is a multi-objective metric which combines the *Power* and the *Training Latency* per epoch (Eq. 4 in Table 4). When we need to consider the total training energy consumption with regards to several epochs, we use the energy consumption per epoch multiplied by the number of epochs, as the total training energy consumption.

TABLE 6. Preliminary results of accuracy of patch sizes in ViT-Ti/16/32/56/112 on 300 epochs (on Laptop using CIRAR-10).

Framework \ Patch Size	ViT-Ti/16	ViT-Ti/32	ViT-Ti/56	ViT-Ti/112
TensorFlow	81.44 (0.0%)	81.78 (0.4%)	77.37 (-5.0%)	63.09 (-22.5%)
PyTorch	84.67 (0.0%)	82.32 (-2.8%)	74.83 (-11.6)	66.98 (-20.9%)

TABLE 7. Preliminary performance results on frameworks, Number of Epochs, Augmentations and models (on laptop using CIFAR-10 [48]).

	Model	Aug.	Acc. (%)	Ep.	Train time (hours)	Inf. (ms) / (b.s.)	Mem. (MB)	ID	AoL
TensorFlow	ViT-Ti/16	No	**80.14**	**100**	**7.3**	61 (32)	3978	14.11	1.31
			80.81	150	10.0	61 (32)	3978	14.11	1.31
			81.44	300	20.0	61 (32)	3978	14.11	1.31
			81.84	500	33.3	61 (32)	3978	14.11	1.31
	ViT-Ti/32	No	**78.27**	**100**	**2.5**	24 (32)	1524	12.92	3.28
			80.09	**150**	**3.8**	24 (32)	1524	13.15	3.34
			81.78	300	7.5	24 (32)	1524	13.43	3.41
			81.75	500	12.5	24 (32)	1524	13.42	3.41
	ViT-Ti/16	Yes	86.01	300	20.0	61 (32)	3894	15.14	1.41
	ViT-Ti/32		82.73	300	7.5	24 (32)	3894	13.58	3.45
PyTorch	ViT-Ti/16	No	84.67	300	19.6	145 (128)	5450	15.34	0.58
	ViT-Ti/32		**82.32**	300	4.3	41 (128)	1274	13.88	2.01
	ViT-Ti/16	Yes	95.55	300	19.6	145 (128)	5450	17.31	0.66
	ViT-Ti/32		92.78	300	4.3	41 (128)	1274	15.65	2.26
	ViT-Li7/4	No	83.59 [25]	300	2.5	25 (128)	909	22.47	3.34
	ViT-Li7/4	Yes	93.57 [25]	300	2.5	25 (128)	909	25.15	3.74
	ViT-Li12/4	No	82.53	300	4.8	46 (128)	1542	15.43	1.79
	ViT-Li12/4	Yes	94.31	300	4.8	46 (128)	1542	17.63	2.05

IV. PRELIMINARY RESULTS: ACCURACY ISSUES ON SMALL DATASETS

Generally, it has been studied that smaller patches provide higher accuracy with longer token lengths, though at a greater computational cost [39]. However, practically speaking, a local optimum patch size is also dependent on the input image resolution, patch sizes [25], and image characteristics [38] (e.g., simple or complex images).

Comparison of Patch Size Effects: Firstly, in order to investigate the effects of the input patch size in accuracy, which is one of most important performance metrics, we measure and compare the accuracies of ViT-Ti/16/32/56/112 using CIFAR-10 dataset without augmentation on 300 epochs, which are based on the reference work [25], with observations of a sufficient number of training epochs on small datasets. While the accuracies of ViT-Ti/16 and ViT-Ti/32 are almost similar on TensorFlow (a comparable degradation of -2.8% on PyTorch), the accuracies of ViT-Ti/56 and ViT-Ti/112 have significant accuracy degradations of -5.0% and -22.5% on TensorFlow (-11.6% and -20.9% on PyTorch), respectively. Therefore, we select ViT-Ti/16 and ViT-Ti/32 for the comprehensive patch and model size characterization for efficient-ViTs.

Comparison of Number of Epoch Effects: Secondly, in order to investigate the effects of the candidate input patch sizes and the number of epochs in the training phase, we analyze basic DA and DR metrics, as shown in Table 7, using the CIFAR-10 dataset without augmentation on TensorFlow. In 100 epochs, ViT-Ti/16 achieves a higher accuracy of 80.14% with a training time of 6.7 hours, compared to 78.27% in ViT-Ti/32 with a reduced training time of 2.5 hours. Additionally, by increasing the number of epochs to 150, ViT-Ti/32's accuracy has increased up

to 80.09% with a training time of 3.8 hours. Finally, the accuracies in 300 and 500 epochs are almost similar between the two models; while ViT-Ti/32 is a little bit more accurate than ViT-Ti/16 in 300 epochs, it is vice versa in 500 epochs. The patterns are clearly shown in the blue triangle (ViT-Ti/32 TF no Aug.) and blue circle (ViT-Ti/16 TF no Aug.) lines in Figure 4, which represent values from Table 7. Figure 4 shows that by increasing the number of epochs, the accuracy gradually increases, and then converges (or subtly decreases due to overfitting [49]).

Furthermore, in terms of training time (the x-axis in Figure 4), it can be inferred that the training time of the ViT-Ti/32 models is almost 3 times faster than that of ViT-Ti/16 models for the same number of epochs. We also observe that models with different patch sizes could achieve similar accuracy by adjusting the number of epochs, maintaining not only a lower training time but also a lower memory usage (Table 7). For the effects of the number of epochs on ViT-Ti/16 and ViT-Ti/32 using small datasets like CIFAR-10/100 without augmentations, we conclude that 300 epochs could be deployed with the related work [25]. Alternatively, our results imply that a more reduced number of epochs could also be considered, according to the accuracy degradation levels in the reduced numbers of epochs.

Comparison of Augmentation Effects: In addition to the effects of the number of epochs, we investigate the effects of augmentation by fixing deployable 300 epochs on both frameworks TensorFlow (TF) and PyTorch (PT). From the perspective of no augmentation between TF and PT frameworks, the ViT-32 TF no Aug. of 81.78% (the third triangle of the blue line ▲ in Figure 4) has almost similar accuracy, compared to ViT-32 PT no Aug. of 82.32% (the red triangle ▲ in Figure 4). From the perspective of augmentation techniques, TF Aug. versions have significantly lower accuracies of 86.01% in ViT-Ti/16 TF w/Aug. (the center green circle ●) and 82.73% in ViT-Ti/32 TF w/Aug. (the bottom-left green triangle ▲) respectively, while PT w/Aug. versions achieve high accuracies of 95.55% in ViT-Ti/16 PT w/Aug. (the top right violet circle ●) and 92.78% in ViT-Ti/32 PT w/Aug. (the top left violet triangle ▲) respectively. This difference arises from the use of different augmentation techniques in the TensorFlow and PyTorch implementation. In detail, while the PT-based codes utilize advanced Mixup [50], CutMix [51], Random erasing [52], and additional random augmentation techniques from the *timm* library [53], the TF-based codes utilize manually implemented naive Mixup and CutMix.

Comparison to ViT-Lite models [25]: Lastly, we compare with pure VIT-Lite models among all the efficient ViT versions [25], although more state-of-the-art efficient ViTs such as CVT and CCT models (i.e., using more advanced training techniques of sequence pooling (SP) rather than class tokenization (CT) and hybrid ViT+CNN models) are included. In addition to the original ViT-Li7/4 models (7 layers with 4×4 patch size from 32×32 images) in the paper [25], we also modify and compare with the ViT-Li12/4

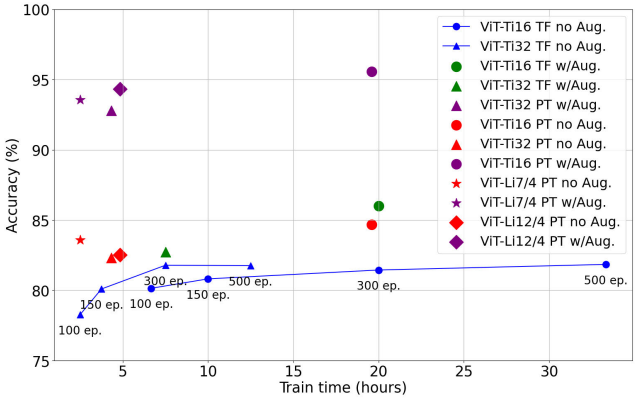


FIGURE 4. Accuracy and train time trade-offs on ViTs(Based on Table 7).

models for fair comparison, considering the same 12 attention layers. While ViT-Li7/4 without and with augmentation versions have accuracies of 83.59% (the bottom-left red asterisk ★) and 93.57% (the top-left violet asterisk ★) respectively, ViT-Li12/4 without and with augmentation versions have accuracies of 82.53% (the bottom-left red diamond ◆) and 94.31% (the top-left violet diamond ◆) respectively.

Note that ViT-Ti32 PT (no Aug. ▲ and w/Aug. ▲) and ViT-Li12/4 (no Aug. ◆ and w/Aug. ◆) have almost similar number of patches, although the number of patches of 64 in ViT-Li12/4 is a little bit more than 49 of ViT-Ti/32. We speculate that ViT-Li12/4 models yield a little bit better results than ViT-Ti/32 due to the larger number of patches and the more advanced sequence pooling (SP) technique, as opposed to the default class tokenization (CT) technique [25].

Summary: Among the baseline models (i.e., ViT-Ti/16/32 no Aug.), the accuracies are almost similar between TF and PT frameworks, while different types of augmentation techniques have more various impacts. Therefore, for further measurements and analysis sections, we utilize the baseline versions without augmentations despite low accuracies, excluding various effects of advanced augmentation techniques. However, our hierarchical and local method (HelLo) can be applied to both more advanced augmentation techniques and more efficient ViT models for higher accuracy (up to ViT+CNN hybrid models by increasing hyper-parameters), using the quantitative metrics for multiple objectives.

V. RESULTS AND ANALYSIS

This section systematically describes a detailed analysis of the results and presents the findings from our experiments. First, we explore the impacts of the patch sizes and the four key hyper-parameters for model sizes: L, D, H, and M. Each subsection focuses on each factor, providing quantitative comparisons and highlighting the trade-offs observed in accuracy, resource usage, and efficiency metrics. We also summarize the effects across different devices (Laptop and Edge), comparing results in device-related (DR) metrics. The analysis provides a summary of key findings on patch

TABLE 8. Results of input patch size effects in ViT-Ti/16 and ViT-Ti/32 on Laptop and Edge. (NOTE: ↑ means the greater the better, and ↓ means the lower the better). Bold is the superior value with the percentage of improvements, compared to the results of ViT-Ti/16. Model sizes are the default for both patch sizes, i.e., L (12), D (192), H (3), and M (768).

P.	Ep.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time/5ep. (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
32	150	-0.05% (80.09)	-7.2% (6.09)	**77.8%** (0.28)	-6.9% (13.14)	**9.2%** (73.8)	**60.7%** (24)	**46.0%** (3.94)	**61.7%** (1524)	**71.2%** (78)	**67.5%** (1.23)	**22.8%** (3962)	**19.1%** (7117)	**155.0%** (3.34)	**243.3%** (1.03)	**73.7%** (6295)	**329.2%** (144.26)
16	**100**	**0.0%** (80.14)	**0.0%** (5.68)	0.0% (1.26)	**0.0%** (14.11)	0.0% (67.6)	0.0% (61)	0.0% (7.30)	0.0% (3978)	0.0% (271)	0.0% (3.78)	0.0% (5133)	0.0% (8798)	0.0% (1.31)	0.0% (0.30)	0.0% (23975)	0.0% (33.61)

and model size effects, helping readers understand which hyper-parameters have the most influence on model performance across the quantitative metrics. Finally, an ablation study is included to validate the combinations of locally optimal hyper-parameters.

A. INPUT PATCH SIZE EFFECTS

In this subsection, we compare and analyze the results of the baseline ViT-Ti/16 (trained for 100 epochs) and ViT-Ti/32 (trained for 150 epochs). As shown in the Table 8, the baseline ViT-Ti/16 (trained for 100 epochs) and ViT-Ti/32 (trained for 150 epochs) have almost the same accuracies of 80.14% and 80.09%, respectively (i.e., only 0.05% difference), while the baseline ViT-Ti/16 (trained for 150 epochs) has almost similar accuracy of 80.81% (only 0.89% less than ViT-Ti/32 on 150 epochs), but with 37% increased training time than ViT-Ti/16 on 100 epochs (i.e., 7.3 vs. 10.0 hours) in Table 7.

DA-Sin. (Acc., Params, MACs): Although the accuracy (80.09% vs. 80.14%) and the number of parameters (6.09M vs. 5.68M) are quite similar between ViT-Ti/16 and ViT-Ti/32, there is a substantial difference in the number of MACs (0.28G vs. 1.26G), with ViT-Ti/32 requiring 4.5x fewer MAC operations than ViT-Ti/16.

The MACs difference between the two models can be explained with the help of Table 5, in which by changing the patch size (P), the number of patches (N) in ViT-Ti/32 can be reduced to one-fourth (1/4) of ViT-Ti/16, while the hidden dimension size (D), MLP size (M), and the number of layers (L) are the same.

DA-Mul. (ID, NetScore): For ID, ViT-Ti/32 has a −6.9% degradation (13.14 vs. 14.11) in accuracy per million parameters compared to ViT-Ti/16 due to the slightly increased (7.2%) number of parameters which results from the increased patch size (P), while the NetScore (accuracy per Params × MACs) achieves 9.2% improvements due to the significantly reduced number of MACs. NetScore metric is more important than ID from the perspective that they have different results when we compare with the inference and train latencies and memory usage. While NetScore can directly reflect latency and memory improvements, ID is not directly related to latency and memory improvements.

DR-Sin. (Inference-/Train-Latency, Memory Usage, Power): Note that DR metrics have different results on different platforms. On Laptop, improvements of 60.7%, 46.0%, and 61.7% in ViT-Ti/32 are achieved in the inference latency, training latency, and GPU memory usage respectively, while on Edge improvements of 71.2%, 67.5%, 22.8%, and 19.1% are achieved in the inference latency, training latency, memory usage, and power consumption respectively. (Note that the memory usage on Edge is the integrated memory, and the power consumption is measured only on Edge, described in Table 4). Interestingly, the improvements of inference latency and training latency (60.7%, 46.0%, and 71.2%, 67.5% on Laptop and Edge respectively) and GPU memory improvement on Laptop (61.7%) are closely related to the device-agnostic MACs improvement of 77.8%. Therefore, if we need to use a proxy DA metric for model efficiency instead of DR metrics on real platforms, the MACs metric is more important than the number of parameters (Params).

DR-Mul. (AoL, Energy, FoM): On Laptop, AoL is improved by 155.0%, which results from the significantly reduced inference latency (24 ms). On Edge, improvements of 243.3%, 73.7%, and 329.2% in AoL, Energy, and FoM respectively are achieved. (Note that energy consumption and FoM are obtained only on Edge due to the availability of power logs). Regarding the power consumption patterns, they will be discussed from the section of layer effects, because they have dynamic patterns even in gradual workload variations. It also makes sense because the power consumption is dependent on dynamic voltage frequency scaling (DVFS) techniques. The improvements of energy consumption, which is a product of time and power, are more related to the improvements of training time rather than the little variations of power consumption (i.e., 7117mW vs. 8798mW).

Key Findings: In terms of the input patch sizes of 16 and 32, 1) firstly, we observe that the ViT-Ti/32 needs more epochs to achieve a comparable accuracy, but with less total training time than ViT-Ti/16. 2) Secondly, among the DA metrics, MACs and NetScore rather than Params and ID can be used as proxy metrics for latency-efficiency and memory-efficiency instead of device-related (DR) metrics on real platforms. 3) Thirdly, energy consumption is more dependent on latency, rather than power consumption on ViT workloads for image classification. Lastly, these key findings conclude that ViT-Ti/32 with longer training epochs (a bigger patch size with a reduced number of tokens) can be a better choice for mobile and edge platforms under the three constraints.

TABLE 9. Results of number of Layers in ViT-32 and ViT-16 on Laptop and Edge, Batch size (bs): 32 on Laptop and 16 on Edge. Baseline: L:12 D:192 H:3 M:768. (red > 2% accuracy drop; yellow - acceptable); best within acceptable < 2% drop and worst in all).

P.	L.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
32	2	-8.04%	73.0%	80.4%	233.3%	14.8%	70.8%	68.3%	50.4%	52.6%	78.2%	24.8%	0.3%	208.4%	89.6%	78.3%	90.3%
		(72.05)	(1.65)	(0.06)	(43.80)	(84.7)	(7)	(1.25)	(756)	(37)	(0.27)	(2978)	(7093)	(10.29)	(1.95)	(1366)	(274.54)
	4	-2.06%	58.4%	64.3%	134.2%	10.6%	54.2%	53.6%	50.4%	56.4%	68.1%	14.4%	-17.7%	112.6%	123.5%	62.5%	89.9%
		(78.03)	(2.54)	(0.10)	(30.78)	(81.6)	(11)	(1.83)	(756)	(34)	(0.39)	(3390)	(8378)	(7.09)	(2.30)	(2362)	(273.93)
	6	-0.91%	**43.8%**	**48.2%**	**76.0%**	**7.0%**	**41.7%**	**42.9%**	**33.6%**	**43.6%**	**56.0%**	**13.2%**	-19.1%	**69.5%**	**73.5%**	**47.6%**	**47.1%**
		(79.18)	(3.42)	(0.15)	(23.13)	(79.0)	(14)	(2.25)	(1012)	(44)	(0.54)	(3438)	(8479)	(5.66)	(1.80)	(3300)	(212.24)
	8	0.29%	29.2%	32.1%	41.8%	4.4%	25.0%	28.9%	**33.6%**	30.8%	43.2%	12.0%	-19.5%	33.8%	45.0%	32.1%	21.3%
		(80.38)	(4.31)	(0.19)	(18.63)	(77.1)	(18)	(2.80)	(1012)	(54)	(0.70)	(3487)	(8507)	(4.47)	(1.49)	(4274)	(174.98)
	10	**0.52%**	14.6%	16.1%	17.9%	2.1%	12.5%	15.5%	0.0%	12.8%	17.6%	0.1%	-1.6%	15.0%	15.5%	16.2%	13.6%
		(80.61)	(5.20)	(0.24)	(15.49)	(75.4)	(21)	(3.33)	(1524)	(68)	(1.01)	(3957)	(7231)	(3.84)	(1.19)	(5272)	(163.94)
	12	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	**0.0%**	0.0%	0.0%	0.0%	0.0%
		(80.09)	(6.09)	(0.28)	(13.14)	(73.8)	(24)	(3.94)	(1524)	(78)	(1.23)	(3962)	(7117)	(3.34)	(1.03)	(6295)	(144.27)
16	2	-4.7%	78.3%	81.0%	334.6%	17.1%	78.7%	80.6%	74.6%	78.2%	82.4%	36.8%	1.5%	342.0%	332.7%	82.6%	339.1%
		(75.49)	(1.23)	(0.24)	(61.32)	(86.79)	(13)	(1.42)	(1012)	(59)	(0.67)	(3245)	(8668)	(5.81)	(1.28)	(4167)	(147.61)
	4	-2.2%	62.7%	64.3%	160.5%	11.2%	62.3%	55.9%	61.7%	60.5%	64.6%	31.1%	5.2%	158.1%	146.4%	66.4%	159.9%
		(77.98)	(2.12)	(0.45)	(36.77)	(80.40)	(23)	(3.22)	(1524)	(107)	(1.34)	(3538)	(8341)	(3.39)	(0.73)	(8046)	(87.37)
	6	-2.2%	47.0%	48.4%	83.5%	6.9%	47.5%	47.2%	35.9%	47.2%	52.0%	22.0%	-3.4%	85.4%	84.4%	50.4%	78.3%
		(77.96)	(3.01)	(0.65)	(25.90)	(79.26)	(32)	(3.87)	(2548)	(143)	(1.82)	(4004)	(9094)	(2.44)	(0.55)	(11894)	(59.95)
	8	-1.1%	**31.3%**	**32.5%**	**43.6%**	**4.2%**	**31.1%**	**32.7%**	**35.9%**	**30.3%**	**32.6%**	**16.0%**	**2.2%**	**43.2%**	**41.4%**	**34.1%**	**44.6%**
		(79.04)	(3.90)	(0.85)	(20.27)	(77.20)	(42)	(4.92)	(2548)	(189)	(2.55)	(4309)	(8604)	(1.88)	(0.42)	(15808)	(48.61)
	10	-0.5%	15.7%	16.7%	17.8%	1.9%	16.4%	16.4%	0.0%	13.3%	15.9%	0.2%	1.9%	18.9%	14.6%	17.5%	16.9%
		(79.64)	(4.79)	(1.05)	(16.63)	(75.51)	(51)	(6.12)	(3978)	(235)	(3.18)	(5125)	(8627)	(1.56)	(0.34)	(19769)	(39.28)
	12	**0.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.14)	(5.68)	(1.26)	(14.11)	(74.12)	(61)	(7.30)	(3978)	(271)	(3.78)	(5133)	(8798)	(1.31)	(0.30)	(23975)	(33.61)

B. NUMBER OF LAYERS EFFECTS

We firstly analyze the number of layer effects on ViT-Ti/32, secondly compare with the effects on ViT-Ti/16, and then summarize key findings of the layer effects.

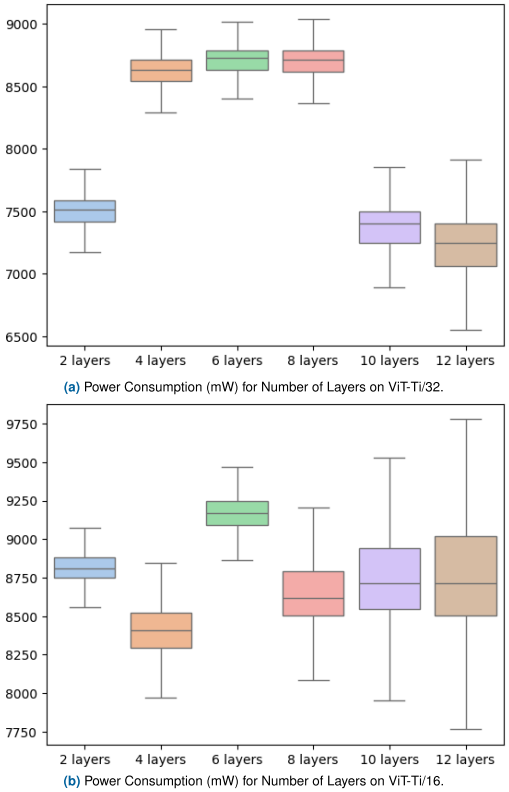
1) RESULTS OF ViT-Ti/32

As shown in Table 9, the overall patterns reveal trade-offs between accuracy and other performance metrics. In other words, greater accuracy degradation leads to more significant improvements in most metrics, except for power consumption, which will be covered separately in detail. In the analyses below, we focus on the results within a 2% accuracy drop (i.e. the 6 layers and above), in line with the design goal of minimal k% accuracy degradation. Note that the baseline is the default ViT-Ti/32 model of 12 layers.

DA-Sin. (Acc., Params, MACs): The accuracies between 8 and 12 layers are almost similar, having the highest accuracy in 10 layers (0.52% higher than the default). The 6 layers, with an accuracy degradation of 0.91%, have improvements of 43.8% in Params and 48.2% in MACs, respectively.

DA-Mul. (ID, NetScore): Due to the improvements in both Params and MACs, the 6 layers also achieve improvements of 76.0% in ID and 7.0% in NetScore, respectively.

DR-Sin. (Inference-/Train-Latency, Memory Usage, Power): On Laptop, inference latency, training latency, and GPU memory usage improved by 41.7%, 42.9%, and 33.6%, respectively. On Edge, inference latency, training latency, and GPU memory usage improved by 43.6%, 56.0%, and 13.2%, respectively, but the power consumption increases by 19.1%. As it was observed in the patch size effects, the inference and training latency improvements both on Laptop and Edge are proportional to the improvements in MACs for all the layers. This shows that the MACs metric is very important both for inference latency and training time.

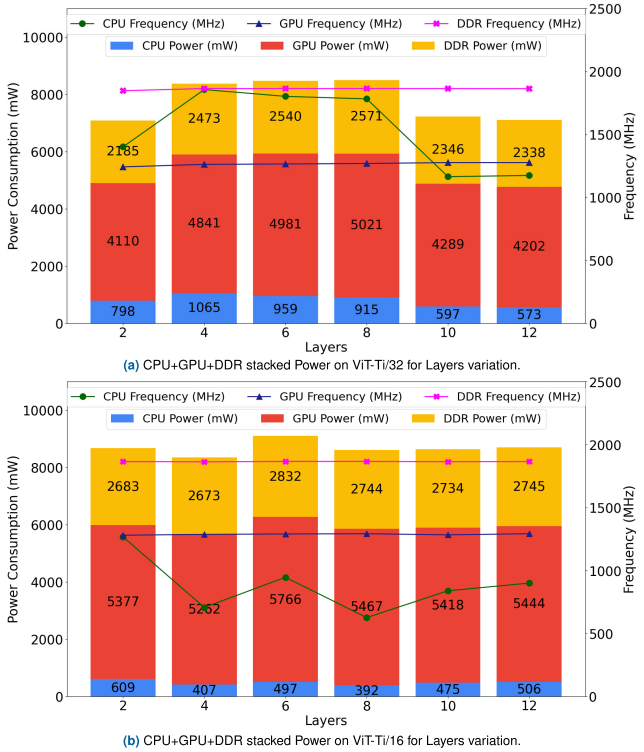


(a) Power Consumption (mW) for Number of Layers on ViT-Ti/32.

(b) Power Consumption (mW) for Number of Layers on ViT-Ti/16.

FIGURE 5. Power consumption in different layers.

Figure 5a shows that the power consumption in ViT-Ti/32 generally tends to vary within a certain level (from 7100 mW



(a) CPU+GPU+DDR stacked Power on ViT-Ti/32 for Layers variation.

(b) CPU+GPU+DDR stacked Power on ViT-Ti/16 for Layers variation.

FIGURE 6. Layers effects: Power and frequencies details.

to 8750 mW) without any linear or specific pattern regardless of the increasing number of layers. Moreover, we provide the stacked average power consumption of CPU, GPU, and memory power consumption in Figure 6a, in which GPU power consumption is dominant, and the memory power consumption is almost constant across different number of layers.

Furthermore, we observe relatively high power consumption between 4 and 8 layers and low power consumption in 2, 10, and 12 layers; and we speculate that the power consumption patterns are quite different from each other in different number of layers due to the dynamically changing CPU, GPU, and Memory frequencies. Table 10 and Figure 6a show the dynamically changing CPU, GPU, and Memory average frequencies from DVFS governors using *tegrastats*. Note that the frequency scaling scheme of CPU, GPU, and Memory governors on TX2 are based on separate CPU and GPU and Memory utilizations respectively [54]; and we speculate that the power consumption patterns of layer effects in ViT-Ti/32 are more related to diverse CPU frequencies rather than relatively similar GPU and Memory frequencies.

DR-Mul. (AoL, Energy, FoM): In terms of AoL, the improvements on Laptop and Edge are 69.5% and 73.5%, respectively. However, the absolute values on the laptop and edge devices differ significantly, at 5.66 and 1.80 respectively, due to the device-related inference latencies, with the laptop being faster than the edge device. Energy consumption per epoch on the edge has an improvement of 47.6% due to the latency improvement of 56%. Since energy consumption is the product of latency and power consumption, the

TABLE 10. Layers effects: Average CPU, GPU, and memory frequencies and utilizations for ViT-Ti/32 and ViT-Ti/16 on Edge (Nvidia TX2).

	CPU freq. (GHz)		CPU util. (%)		GPU freq. (GHz)		GPU util. (%)		DDR freq. (GHz)	
#Layers	32	16	32	16	32	16	32	16	32	16
2	1.403	1.266	62	37	1.244	1.279	64	84	1.849	1.864
4	1.858	0.706	54	49	1.264	1.286	77	83	1.866	1.862
6	1.804	0.945	53	39	1.267	1.289	81	89	1.866	1.865
8	1.784	0.625	53	49	1.271	1.291	81	85	1.866	1.865
10	1.166	0.838	62	49	1.278	1.282	68	85	1.866	1.863
12	1.176	0.899		53	1.278	1.291	67	83	1.865	1.864

observed energy improvements can be attributed to latency improvements rather than to power consumption changes, given that power variations (ranging from 7100 mW to 8750 mW) are relatively small compared to the changes in training time. Lastly, the multi-objective FoM improvement on the edge in 6 layers reaches 47.1%, which can be attributed to the inference time improvement (43.6%), the accuracy drop (0.91%), and the power increase (19.1%).

2) RESULTS OF ViT-Ti/16 COMPARED TO ViT-Ti/32

The overall improvement levels of ViT-Ti/16 are similar yet lower than those of ViT-Ti/32, except for a few important metrics, such as accuracy and power consumption. However, the absolute values (in parentheses) for the two models largely differ, as already observed in the patch size effects.

DA-Sin. (Acc., Params, MACs): The accuracy degradation is more serious in ViT-Ti/16 (-2.2% in 6 layers, compared to -0.91% in 6 layers of ViT-Ti/32). Within the same 2% threshold in accuracy degradation, the accuracy of 8 layers satisfies with -1.1% degradation. Hence, we compare the results of 8 layers in ViT-Ti/16 with those of 6 layers in ViT-Ti/32 focusing on the best improvements (in bold) and the absolute values (in parentheses).

When we compare the improvements in MACs between the two models, ViT-Ti/32 shows higher relative improvements (48.2% vs. 32.5%) due to more serious accuracy degradation in ViT-Ti/16, decreasing MACs from 0.28G (baseline of 12 layers) to 0.15G (6 layers). For reference, the ViT-Ti/16 model shows a reduction from 1.26G (baseline of 12 layers) to 0.85G (8 layers). In terms of the absolute values (0.15G vs. 0.85G), the ViT-Ti/16 of 8 layers has 5.67× more MACs than ViT-Ti/32 of 6 layers.

DA-Mul. (ID, NetScore): The pattern of ID and NetScore are almost proportional to those of MACs.

DR-Sin. (Inference-/Train-Latency, Memory Usage, Power): The patterns of inference and train latencies are almost similar to that of MACs. Regarding the device-related comparisons, the Laptop is much faster than the Edge in terms of inference latency (3-5×) and train time (5-9×). While memory usage improvements are almost similar, the power consumption patterns are quite different between ViT-Ti/16 layers and ViT-Ti/32 layers due to the dynamically scaling frequencies (Figure 6 and Table 10).

Furthermore, we compare and analyze CPU and GPU utilizations in Table 10. Through all layers, ViT-Ti/32 and ViT-Ti/16 have ranges of 53%–62% and 37%–53%

TABLE 11. Results of D size in ViT-32 and ViT-16 on Laptop and Edge, Batch size: 32 on Laptop and 16 on Edge Baseline: L:12 D:192 H:3 M:768. (red > 2% drop; yellow - acceptable); best within acceptable < 2% drop and worst in all).

P.	D.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
32	96	0.13%	**57.1%**	**57.7%**	**133.6%**	**10.1%**	**12.5%**	**14.3%**	0.0%	**28.2%**	**23.3%**	3.26%	**14.7%**	**14.5%**	**39.5%**	**34.6%**	**63.6%**
		(80.22)	(2.61)	(0.12)	(30.70)	(81.24)	(21)	(3.38)	(1524)	(56)	(0.94)	(3833)	(6070)	(3.82)	(1.43)	(4116)	(236.00)
	128	**0.78%**	39.9%	40.2%	68.0%	6.3%	8.3%	10.1%	0.0%	21.8%	15.4%	**3.72%**	7.7%	10.2%	29.1%	22.0%	39.9%
		(80.87)	(3.66)	(0.17)	(22.08)	(78.42)	(22)	(3.54)	(1524)	(61)	(1.04)	(3815)	(6567)	(3.68)	(1.33)	(4913)	(201.86)
	192	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.09)	(6.09)	(0.28)	(13.14)	(73.81)	(24)	(3.94)	(1524)	(78)	(1.23)	(3962)	(7117)	(3.34)	(1.03)	(6295)	(141.26)
	256	0.51%	-45.9%	-47.0%	-31.0%	-4.3%	-8.3%	-4.7%	0.0%	-26.9%	-14.7%	2.63%	-4.8%	-8.3%	-20.7%	-20.3%	-24.4%
		(80.60)	(8.89)	(0.41)	(9.06)	(70.60)	(26)	(4.13)	(1524)	(99)	(1.41)	(3858)	(7462)	(3.10)	(0.81)	(7572)	(109.11)
16	96	-0.62%	**57.6%**	**57.1%**	**134.2%**	**10.8%**	**21.3%**	**24.3%**	0.0%	**32.5%**	**35.3%**	**9.38%**	**3.4%**	**26.1%**	**46.9%**	**37.5%**	**52.0%**
		(79.52)	(2.41)	(0.54)	(33.05)	(74.88)	(48)	(5.53)	(3978)	(183)	(2.45)	(4628)	(8503)	(1.66)	(0.43)	(14994)	(51.10)
	128	-0.40%	40.4%	39.7%	66.9%	6.4%	18.0%	15.9%	0.0%	17.3%	22.8%	7.34%	-0.4%	21.4%	20.4%	22.5%	19.9%
		(79.74)	(3.39)	(0.76)	(23.55)	(71.96)	(50)	(6.14)	(3978)	(224)	(2.92)	(4756)	(8832)	(1.59)	(0.36)	(18574)	(40.31)
	192	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.14)	(5.68)	(1.26)	(14.11)	(67.61)	(61)	(7.30)	(3978)	(271)	(3.78)	(5133)	(8798)	(1.31)	(0.30)	(23975)	(33.61)
	256	**0.81%**	-46.9%	-45.2%	-31.2%	-4.6%	-13.1%	-16.4%	0.0%	-17.3%	-10.0%	1.56%	-8.3%	-10.7%	-13.9%	-19.1%	-20.5%
		(80.95)	(8.34)	(1.83)	(9.71)	(64.49)	(69)	(8.50)	(3978)	(318)	(4.16)	(5053)	(9527)	(1.17)	(0.25)	(28547)	(26.72)

on CPU utilization (64%–81% and 83%–89% on GPU utilization), respectively; more specifically, ViT-32 has more CPU-intensive utilization, while ViT-Ti/16 has more GPU-intensive utilization. Although it is beyond our scope to investigate CPU-GPU optimized utilization perspectives in this work, one clear observation is that there are opportunities in terms of CPU and GPU hardware utilization perspectives.

DR-Mul. (AoL, Energy, FoM): AoL is directly proportional to the inference latency. In terms of absolute values of FoM, ViT-Ti/32 has 4.4× better score compared to ViT-Ti/16 (212.24 vs 48.61) despite similar relative improvements (47.1% vs. 44.6%), even allowing a slightly higher accuracy (79.18 vs. 79.04). Considering that differences in power consumption (8479 mW vs. 8604 mW) and accuracy are small, the FoM improvement in ViT-Ti/32 is primarily due to the reduced inference latency (i.e., 44 ms in ViT-Ti/32 vs. 189 ms in ViT-Ti/16). By the same token, the difference in absolute values of the energy consumption (i.e., 4.8× better, 3300 J vs. 15808 J) can be explained by the training time.

Key Findings: 1) Firstly, the accuracy degradation/trade-off is more serious in ViT-Ti/16 compared to ViT-Ti/32. This is related to the challenging and intrinsic ViT workloads of the high quadratic computation (i.e., according to N, number of patches/tokens) and memory requirements [1]. Note that the N of Ti/16 is four times that of Ti/32.

2) Secondly, the pattern of power consumption is not dependent on (or proportional to) the number of layers (ViT workloads) but rather on separate frequencies of DVFS governors, with GPU power accounting for 55-60% of the total power consumption. 3) Thirdly, in terms of absolute values, the energy consumption and FoM of ViT-Ti/32 achieve 4.4× and 4.8× better results than those of ViT-Ti/16, with only a negligible 0.91% accuracy degradation. These key findings show that the ViT-Ti/32 with the reduced number of layers could be an alternative for the efficient-ViT models on small datasets.

C. EFFECTS OF D SIZE

In this section, we investigate how D size (not D + MLP) affects the quantitative metrics. To do this, we change only D sizes, keeping the same MLP size of 768 (the default baseline) by modifying the *mlp_rate* in our code (e.g., 96 × 8 (*mlp_rate*) = 768).

1) RESULTS OF ViT-Ti/32

DA-Sin. (Acc, Params, MACs): Interestingly, even the smallest D size of 96 results in 0.13% accuracy improvements, with a better 0.78% accuracy improvement in a D size of 128 (Table 11). Additionally, the improvements in Params and MACs reach 57.1% and 57.7%, respectively, surpassing the improvements observed from layer effects in terms of Params and MACs. However, the improvements in inference latency and training time are lower than those observed from layer effects, which we further investigate in the corresponding DR metrics.

DA-Mul. (ID, NetScore): Due to the improvements of DA-Sin. metrics, ID and NetScore are also greatly improved by 133.6% and 10.1% respectively. (Note that the DR-Mul. improvements are the highest among all the model size effects).

DR-Sin. (Inference-/Train-Latency, Memory Usage, Power): On Laptop, the improvements of 12.5% and 14.3% in inference latency and train time are achieved, respectively, while the edge device has the improvements of 28.2% and 23.3% in inference latency and train time, respectively. More specifically, on Edge, despite substantial improvements in Params and MACs (57.1% and 57.7%), inference latency and train time improvements (28.2% and 23.3%) are lower than those of the layer effects (43.6% and 56.0%) in ViT-Ti/32 (Table 9). This shows that the number of MACs still has limitations in evaluating the inference/train latency, although it can be used as an alternative proxy metric.

In terms of memory usages on ViT-Ti/32, Laptop GPU memory usages remain constant in spite of the D size

variations, while the integrated memory usages on Edge change slightly, having improvements within 4%. From the perspective of power consumption for ViT-Ti/32 shown in Figure 7a, D size of 96 achieves the highest improvement of 14.7% and the power consumption grows gradually as the D size increases (from 6250 mW up to 7500 mW). In order to speculate reasons for the power and memory patterns, we investigate the CPU, GPU and Memory power and frequencies breakdown presented in Figure 7 and Table 12. We observe that the memory (i.e., DDR) power consumption is almost constant (less than 1% difference between 2206 and 2421 mW) and its frequency also is almost the same (1.86 GHz) across all D size values. We speculate that the CPU workload reduces as D size grows (decreasing CPU frequency line ●), whereas GPU workload grows (slightly increasing GPU frequency line ▲). In the meantime, memory usage is almost constant (almost horizontal frequency line ■).

DR-Mul. (AoL, Energy, FoM): AoL is improved by 14.5% and 39.5% on Laptop and Edge, respectively, which happen to be proportional to the improvements of inference latency. On Edge, the improvements of 34.6% in energy consumption and 63.6% in FoM score (236.0) are achieved due to the improvements in power, inference latency, and even accuracy.

2) RESULTS OF ViT-Ti/16 COMPARED TO ViT-Ti/32

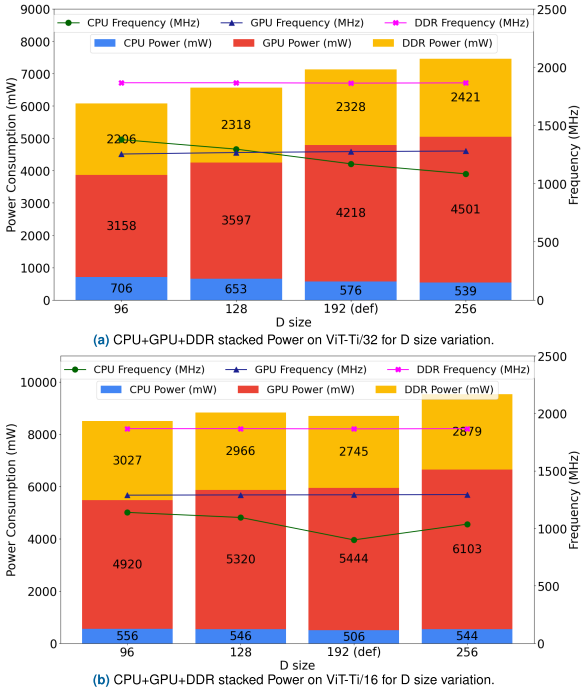
Overall improvement patterns of ViT-Ti/16 are similar to those of ViT-Ti/32 with the best results in D size of 96.

DA-Sin. (Acc, Params, MACs): ViT-Ti/16 shows a slight accuracy degradation (-0.62%) compared to the ViT-Ti/32 (0.13%) despite having similar Params and MAC improvements.

DA-Mul. (ID, NetScore): The improvements in ID and NetScore of ViT-Ti/16 and ViT-Ti/32 are also almost similar due to similar DA-Sin. improvements.

DR-Sin. (Inference-/Train-Latency, Memory Usage, Power): On Laptop, the improvements of inference latency and train time in ViT-Ti/16 (21.3% and 24.3%) are higher than those of ViT-Ti/32 (12.5% and 14.3%). On Edge, the inference latency (32.5%) and train time (35.3%) improvements of ViT-Ti/16 are higher than those of ViT-Ti/32 (28.2% and 23.3%, respectively). The pattern of memory usage is constant on the Laptop and showcases relatively little variations (up to 10%) on the Edge. Also, ViT-Ti/16 has higher power consumption levels ranging from 8500 mW to 9500 mW in Figure 7b with higher GPU frequencies (i.e., mostly near to the maximum 1.3 GHz in Table 12), compared to the levels of ViT-Ti/32 from 6250 mW to 7500 mW in as shown in Figure 7a.

Similar to the layer variations, we present the average results of CPU and GPU utilizations on D size effects in Table 12. Through all the D size variations, ViT-Ti/32 and ViT-Ti/16 have the ranges of 60%–68% and 47%–53% on CPU utilization (53%–68% and 83%–90% on GPU utilization) respectively, resulting in the same pattern



(a) CPU+GPU+DDR stacked Power on ViT-Ti/32 for D size variation.

(b) CPU+GPU+DDR stacked Power on ViT-Ti/16 for D size variation.

FIGURE 7. D size: Power and frequencies details.

TABLE 12. D size effects: Average CPU, GPU, and DDR frequencies and utilizations for ViT-Ti/32 and ViT-Ti/16 on Edge (Nvidia TX2).

	CPU freq. (GHz)		CPU util. (%)		GPU freq. (GHz)		GPU util. (%)		DDR freq. (GHz)	
D size	32	16	32	16	32	16	32	16	32	16
96	1.377	1.138	68	47	1.253	1.288	53	86	1.865	1.866
128	1.295	1.094	66	48	1.266	1.290	62	89	1.865	1.866
192	1.169	0.899	62	53	1.274	1.291	67	83	1.862	1.864
256	1.083	1.036	60	49	1.279	1.293	68	90	1.864	1.866

of CPU-intensive utilization in ViT-32 and GPU-intensive utilization in ViT-Ti/16.

Based on the observations and comparisons, we speculate that the challenging device-related differences may be attributed to the differences in NVIDIA GPU specifications of the number of Streaming MultiProcessors (SM) and the number of CUDA cores per SM, but with the same underlying CUDA parallel programming (i.e., the same kernel configurations for the number of blocks and the number of threads per block) in the framework of TensorFlow. Specific reasons need to be more investigated using NVIDIA metrics of profilers like *nvprof,bib:55*, which is beyond scope of the study and will be a future work.

DR-Mul. (AoL, Energy, FoM): Both on Laptop and Edge, the AoL scores are improved by 26.1% and 46.9% respectively. However, in terms of absolute AoL values, ViT-Ti/16 has 2-3 times lower (i.e., worse) scores than ViT-Ti/32. Additionally, we observe the improvements of 37.5% in energy consumption and 52.0% in FoM score in ViT-Ti/16; yet FoM improvements in ViT-Ti/16 are lower than ViT-Ti/32 (63.6%) due to the lower improvements in power consumption (i.e., 3.4% vs. 14.7%).

Key Findings: 1) The D size changes do not have an accuracy drop in ViT-Ti/32 and have almost negligible

TABLE 13. Results of Heads size in ViT-32 and ViT-16 on Laptop and Edge, Batch size = 32 on Laptop and 16 on Edge. Baseline: L:12 D:192 H:3 M:768 (red > 2% drop; yellow - acceptable); best within < 2% drop and worst in all).

P.	H.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow. × Inf.) ↑
32	1	-0.76%			-0.95%	-0.22%	**4.2%**	3.8%	0.0%	-2.6%	**0.9%**	-0.1%	**1.2%**	3.36%	-3.4%	**2.1%**	-2.3%
		(79.33)			(13.02)	(73.64)	(23)	(3.79)	(1524)	(80)	(1.22)	(3966)	(7033)	(3.45)	(0.99)	(6165)	(140.99)
	2	0.07%			0.07%	0.02%	**4.2%**	**4.8%**	0.0%	**1.3%**	0.5%	0.0%	0.4%	4.43%	**1.4%**	1.0%	**1.8%**
		(80.15)			(13.15)	(73.82)	(23)	(3.75)	(1524)	(77)	(1.22)	(3962)	(7086)	(3.48)	(1.04)	(6234)	(146.90)
	3	0.00%	6.09	0.28	0.00%	0.00%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.09)			(13.14)	(73.81)	(24)	(3.94)	(1524)	(78)	(1.23)	(3962)	(7117)	(3.34)	(1.03)	(6295)	(144.26)
	4	**0.98%**			**1.22%**	**0.29%**	**4.2%**	2.8%	0.0%	-1.3%	-1.7%	-0.1%	-0.4%	**5.62%**	-0.1%	-2.1%	-0.5%
		(81.07)			(13.30)	(74.02)	(23)	(3.83)	(1524)	(79)	(1.25)	(3965)	(7146)	(3.52)	(1.03)	(6425)	(143.61)
	6	0.80%			0.80%	0.19%	0.0%	0.5%	0.0%	-2.6%	-1.7%	**3.5%**	-18.3%	0.80%	-1.7%	-20.3%	-16.9%
		(80.73)			(13.25)	(73.95)	(24)	(3.92)	(1524)	(80)	(1.25)	(3825)	(8420)	(3.36)	(1.01)	(7573)	(119.86)
16	1	-3.3%			-4.1%	-1.1%	8.2%	9.5%	0.0%	5.2%	7.0%	-2.0%	0.6%	4.4%	1.1%	7.6%	1.7%
		(76.83)			(13.53)	(66.87)	(56)	(6.61)	(3978)	(257)	(3.52)	(5237)	(8745)	(1.37)	(0.30)	(22161)	(34.19)
	2	-1.1%			-1.4%	-0.4%	**6.6%**	**6.7%**	0.0%	**3.7%**	**4.1%**	-1.2%	0.3%	**5.5%**	**2.4%**	**4.4%**	**2.7%**
		(79.04)			(13.92)	(67.37)	(57)	(6.81)	(3978)	(261)	(3.63)	(5196)	(8769)	(1.39)	(0.30)	(22922)	(34.53)
	3	0.0%	5.68	1.26	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	**0.0%**	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.14)			(14.11)	(67.61)	(61)	(7.30)	(3978)	(271)	(3.78)	(5133)	(8798)	(1.31)	(0.30)	(23975)	(33.61)
	4	-0.2%			-0.3%	-0.1%	-4.9%	-4.2%	0.0%	-7.0%	-5.7%	-1.0%	**0.5%**	-5.0%	-6.8%	-5.2%	-6.4%
		(79.90)			(14.07)	(67.55)	(64)	(7.61)	(3978)	(290)	(4.00)	(5184)	(8754)	(1.25)	(0.28)	(25213)	(31.47)
	6	**0.5%**			**0.6%**	**0.2%**	-13.1%	-14.5%	0.0%	-8.9%	-3.7%	-2.6%	-5.4%	-11.1%	-7.6%	-9.3%	-12.3%
		(80.63)			(14.20)	(67.71)	(69)	(8.36)	(3978)	(295)	(3.92)	(5264)	(9274)	(1.17)	(0.27)	(26203)	(29.47)

accuracy drop (0.62%) in ViT-Ti/16 in spite of the highest improvements of Params and MACs among all the model size changes (L, D, H, and M). 2) However, the improvements of inference latency and train time are lower than those of the layer effects (L). 3) D size changes have no or fewer effects on memory usages compared to the layer effects (L). 4) The FoM improvement in D size of 96 on ViT-Ti/32 is the highest among all the model size effects.

D. HEADS NUMBER EFFECT

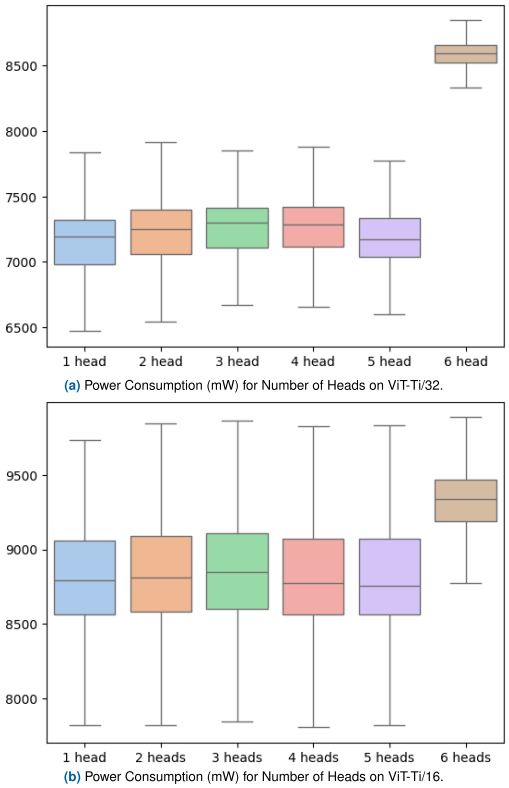
In addition to the default 3 heads, we investigate the effects of the number of heads from 1 to 6. According to the multi-head self-attention (MHSA) mechanism [1], the number of heads can affect the model's performance (by allowing more diverse attention patterns) without changing the number of parameters, and that the MHSA allows parallel computations with the same dimensionality (D) for their combined output.

1) RESULTS OF ViT-Ti/32

As shown in Table 13, changing the numbers of heads does not significantly affect the metrics compared to other effects. Therefore, we provide a brief summary of this section.

DA Metrics: With the constant number of Params and MACs, the accuracy trade-offs are not much significant (only 0.76% accuracy drop even in the single head). The ID and NetScore fluctuations are also minimal.

DR Metrics: In terms of inference latency, while the laptop has 4.2% improvement, whereas the edge device has -2.6% degradation. Due to the increased inference latency on Edge, AoL and FoM also declined by −3.4% and -2.3%, respectively. Regarding the power and energy, as demonstrated in Figure 8a, the pattern is quite stable until the 6 heads.



(a) Power Consumption (mW) for Number of Heads on ViT-Ti/32.

(b) Power Consumption (mW) for Number of Heads on ViT-Ti/16.

FIGURE 8. Power consumption in different number of heads.

2) RESULTS OF ViT-Ti/16 COMPARED TO ViT-Ti/32

DA Metrics: The accuracy degradation is more serious in ViT-Ti/16 (−3.3% vs. −0.76% in ViT-Ti/32), although the number of Params and MACs is the same. In addition, small performance drops are observed in ID and NetScore.

DR Metrics: Improvement patterns of inference latency and train time on both Laptop and Edge are a little bit

TABLE 14. Results of MLP size in ViT-32 and ViT-16 on Laptop and Edge, Batch size = 32 on Laptop and 16 on Edge. Baseline: L:12 D:192 H:3 M:768 (red > 2% drop; yellow - acceptable); best within < 2% drop and worst in all).

P.	M.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow. × Inf.) ↑
32	192	-0.05%	**45.6%**	**46.4%**	**83.7%**	**7.2%**	**20.8%**	**13.2%**	**33.6%**	**25.6%**	**26.9%**	**10.4%**	**2.6%**	**26.2%**	**34.4%**	**28.8%**	**38.1%**
		(80.04)	(3.32)	(0.15)	(24.14)	(79.17)	(19)	(3.42)	(1012)	(58)	(0.90)	(3549)	(6929)	(4.21)	(1.38)	(4482)	(199.18)
	384	0.31%	30.4%	32.1%	44.2%	4.5%	12.5%	7.9%	**33.6%**	12.8%	18.9%	3.8%	1.3%	14.7%	15.1%	20.0%	16.7%
		(80.40)	(4.24)	(0.19)	(18.95)	(77.15)	(21)	(3.63)	(1012)	(68)	(1.00)	(3811)	(7025)	(3.83)	(1.18)	(5038)	(168.30)
	768	0.00%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.09)	(6.09)	(0.28)	(13.14)	(73.83)	(24)	(3.94)	(1524)	(78)	(1.23)	(3962)	(7117)	(3.34)	(1.03)	(6295)	(144.26)
	960	**0.48%**	-15.2%	-17.9%	-12.7%	-1.7%	-16.7%	-11.2%	0.0%	-11.5%	-10.9%	3.3%	-0.3%	-13.8%	-9.8%	-11.3%	-10.1%
		(80.57)	(7.02)	(0.33)	(11.48)	(72.60)	(28)	(4.38)	(1524)	(87)	(1.36)	(3831)	(7140)	(2.88)	(0.93)	(7003)	(129.71)
	1152	0.21%	-30.4%	-32.1%	-23.1%	-3.1%	-20.8%	-16.2%	0.0%	-16.7%	-11.2%	8.8%	-0.1%	-17.0%	-14.1%	-11.3%	-14.1%
		(80.30)	(7.95)	(0.37)	(10.11)	(71.51)	(29)	(4.58)	(1524)	(91)	(1.37)	(3613)	(7124)	(2.77)	(0.88)	(7008)	(123.86)
16	192	-0.2%	**48.9%**	**43.2%**	**95.2%**	**7.9%**	**24.6%**	**27.0%**	**36.8%**	**23.2%**	**19.4%**	**17.8%**	**5.6%**	**32.3%**	**31.9%**	**28.6%**	**39.8%**
		(79.95)	(2.90)	(0.71)	(27.55)	(72.95)	(46)	(5.33)	(2468)	(205)	(2.9)	(4219)	(8303)	(1.74)	(0.39)	(17111)	(46.97)
	384	-0.5%	32.6%	28.8%	47.4%	4.5%	14.8%	18.2%	0%	12.7%	10.2%	0.2%	4.5%	16.5%	15.5%	19.6%	20.9%
		(79.60)	(3.83)	(0.90)	(20.79)	(70.69)	(52)	(5.97)	(3908)	(233)	(3.2)	(5142)	(8405)	(1.53)	(0.34)	(19282)	(40.65)
	768	**0.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.14)	(5.68)	(1.26)	(14.11)	(67.62)	(61)	(7.36)	(3908)	(267)	(3.5)	(5133)	(8798)	(1.31)	(0.30)	(23975)	(33.61)
	960	-0.5%	-16.3%	-14.4%	-14.6%	-2.0%	-8.2%	-16.8%	0%	-14.6%	-16.8%	-1.5%	-0.3%	-8.1%	-12.0%	-9.9%	-12.2%
		(79.65)	(6.61)	(1.44)	(12.06)	(66.27)	(66)	(8.53)	(3908)	(306)	(4.1)	(5208)	(8824)	(1.21)	(0.26)	(26345)	(29.50)
	1152	-0.7%	-32.6%	-28.8%	-25.3%	-3.7%	-18.0%	-22.2%	0%	-23.2%	-26.3%	-1.2%	-1.3%	-16.1%	-18.4%	-20.0%	-19.4%
		(79.39)	(7.53)	(1.62)	(10.54)	(65.13)	(72)	(8.92)	(3908)	(329)	(4.5)	(5197)	(8910)	(1.10)	(0.24)	(28760)	(27.08)

different within 4% improvements, while the memory usages and power consumptions are quite similar. And, all the DR-Mul. metrics are mainly dependent on the improvements of inference latency and train time.

Key Findings: The degree of impact of number of heads on small datasets is the least among all the model changes, along with the observation that ViT-Ti/16 has higher trade-offs between accuracy and the reduced number of heads.

E. MLP SIZE EFFECT

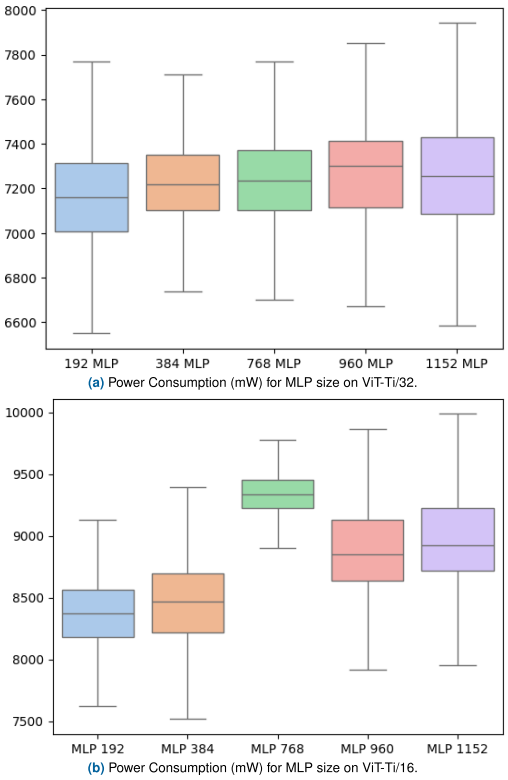
1) RESULTS OF ViT-Ti/32

DA-Sin. (Acc, Params, MACs): By reducing MLP size to 192, Params and MACs are reduced almost by half (45.6% and 46.4% respectively) with a negligible accuracy degradation of 0.05%.

DA-Mul. (ID, NetScore): ID and NetScore also improved by 83.7% and 7.2%, respectively, due to the improvements in DA-Sin. metrics.

DR-Sin. (Inference-/Train-Latency, Memory Usage, Power): Unlike the layer effects, despite the high improvements of Params and MACs, the improvements of inference latency and train time are not high enough, which indicates that they are not directly proportional. For example, the inference latency and train time on Laptop reduced by 20.8% and 13.2% respectively (25.6% and 26.9% respectively on Edge). In terms of memory usage, significant improvements (up to 33.6% on Laptop) are observed on Laptop, along with lower improvements (10.4%) on Edge. The power consumption curve in ViT-Ti/32 is almost constant, as depicted in Figure 9a.

DR-Mul. (AoL, Energy, FoM): On Laptop, the AoL score improved by 26.2%. On Edge, improvements of 34.4% in AoL, 28.8% in energy consumption, and 38.1% in FoM score are observed. Overall, multi-objective improvements, with the exception of FoM, are quite similar to those observed in D size.



(a) Power Consumption (mW) for MLP size on ViT-Ti/32.

(b) Power Consumption (mW) for MLP size on ViT-Ti/16.

FIGURE 9. Power consumption in MLP size.

2) RESULTS OF ViT-Ti/16 COMPARED TO ViT-Ti/32

The MLP size improvements on ViT-Ti/16 are almost similar to ViT-Ti/32, except the train time and the power consumption patterns, on which we focus for brevity.

DA Metrics: All the improvements in metrics are almost equal to those of ViT-Ti/32.

DR Metrics: In terms of the train time, while ViT-Ti/16 (27.0%) has higher improvements than ViT-Ti/32 (13.2%) on Laptop, ViT-Ti/32 (26.9%) has better improvements than

TABLE 15. Optimal variations for Ti32 on laptop and edge (yellow - < 2%; green - < 1% drop); best and worst for each hyper-parameter).

H.	Val.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
#L.	6	-0.91% (79.18)	43.8% (3.42)	48.2% (0.15)	76.0% (23.13)	7.0% (79.0)	**41.7%** (14)	**42.9%** (2.25)	**33.6%** (1012)	**43.6%** (44)	**56.0%** (0.54)	**13.2%** (3438)	-19.1% (8479)	**69.5%** (5.66)	**73.5%** (1.80)	**47.6%** (3300)	47.1% (212.24)
D.	96	**0.13%** (80.22)	**57.1%** (2.61)	**57.7%** (0.12)	**133.6%** (30.70)	**10.1%** (81.24)	12.5% (21)	14.3% (3.38)	0.0% (1524)	28.2% (56)	23.3% (0.94)	3.26% (3833)	**14.7%** (6070)	14.5% (3.82)	39.5% (1.43)	34.6% (4116)	**63.6%** (236.00)
#H.	1	-0.76% (79.33)	0.0% (6.09)	0.0% (0.28)	-0.95% (13.02)	-0.22% (73.64)	4.2% (23)	3.8% (3.79)	0.0% (1524)	-2.6% (80)	0.9% (1.22)	-0.1% (3966)	1.2% (7033)	3.36% (3.45)	-3.4% (0.99)	2.1% (6165)	-2.3% (140.99)
M.	192	-0.05% (80.04)	45.6% (3.32)	46.4% (0.15)	83.7% (24.14)	7.2% (79.17)	20.8% (19)	13.2% (3.42)	**33.6%** (1012)	25.6% (58)	26.9% (0.90)	10.4% (3549)	2.6% (6929)	26.2% (4.21)	34.4% (1.38)	28.8% (4482)	38.1% (199.18)
P32	def	0.0% (80.09)	0.0% (6.09)	0.0% (0.28)	0.0% (13.14)	0.0% (73.80)	0.0% (24)	0.0% (3.94)	0.0% (1524)	0.0% (78)	0.0% (1.23)	0.0% (3962)	0.0% (7117)	0.0% (3.34)	0.0% (1.03)	0.0% (6295)	0.0% (144.26)
P16	def	80.14	5.68	1.26	14.11	67.62	61	7.30	3978	271.5	3.78	5133	8798	1.31	0.30	23975	33.61

TABLE 16. Optimal variations for Ti16 on laptop and edge (yellow - < 2%; green - < 1% drop); best and worst for each hyper-parameter).

H.	Val.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
#L.	8	-1.1% (79.04)	31.3% (3.90)	32.5% (0.85)	43.6% (20.27)	4.2% (77.20)	**31.1%** (42)	**32.7%** (4.92)	35.9% (2548)	30.3% (189)	32.6% (2.55)	16.0% (4309)	2.2% (8604)	**43.2%** (1.88)	41.4% (0.42)	34.1% (15808)	44.6% (48.61)
	10	-0.50% (79.64)	15.7% (4.79)	16.7% (1.05)	17.8% (16.63)	1.9% (75.51)	16.4% (51)	16.4% (6.12)	0.0% (3978)	13.3% (235)	15.9% (3.18)	0.2% (5125)	1.9% (8627)	18.9% (1.56)	14.6% (0.34)	17.5% (19769)	16.9% (39.28)
D	96	-0.62% (79.52)	**57.6%** (2.41)	**57.1%** (0.54)	**134.2%** (33.05)	**10.8%** (74.88)	21.3% (48)	24.3% (5.53)	0.0% (3978)	**32.5%** (183)	**35.3%** (2.45)	9.38% (4628)	3.4% (8503)	26.1% (1.66)	**46.9%** (0.43)	**37.5%** (14994)	**52.0%** (51.10)
#H.	2	-1.1% (79.04)	0.0% (5.68)	0.0% (1.26)	-1.4% (13.92)	-0.4% (67.37)	6.6% (57)	6.7% (6.81)	0.0% (3978)	3.7% (261)	4.1% (3.63)	-1.2% (5196)	0.3% (8769)	5.5% (1.39)	2.4% (0.30)	4.4% (22922)	2.7% (34.53)
M.	192	-0.2% (79.95)	48.9% (2.90)	43.2% (0.71)	95.2% (27.55)	7.9% (72.95)	24.6% (46)	27.0% (5.33)	**36.8%** (2468)	23.2% (205)	19.4% (2.9)	**17.8%** (4219)	**5.6%** (8303)	32.3% (1.74)	31.9% (0.39)	28.6% (17111)	39.8% (46.97)
P16	def	**0.0%** (80.14)	0.0% (5.68)	0.0% (1.26)	0.0% (14.11)	0.0% (67.62)	0.0% (61)	0.0% (7.30)	0.0% (3978)	0.0% (271)	0.0% (3.78)	0.0% (5133)	0.0% (8798)	0.0% (1.31)	0.0% (0.30)	0.0% (23975)	0.0% (33.61)
P32	def	80.09	6.09	0.28	13.14	73.80	24	3.94	1524	78	1.23	3962	7117	3.34	1.03	6295	144.26

ViT-Ti/16 (19.4%) on Edge. Note that the ViT training phase is composed of backward and forward passes, while the inference phase is composed of only forward pass. Due to the difference in computations, the improvements between the two phases are not always proportional and have variability on different devices [56], [57], although it was proportional in the layer effects in Table 9. Therefore, the challenging device-related (DR) investigations between inference and training phases will be considered as our future work.

Unlike the stable and low power consumption in ViT-Ti/32, ViT-Ti/16 has a higher variance and higher power consumption levels up to 9250 mW (median) with narrower spread. According to our observations, the ViT-Ti/16 GPU frequencies are higher (near to the maximum of 1.3 GHz) than those of ViT-Ti/32; and the MLP size of 768 has the highest CPU frequency with the highest power consumption.

Key Findings: 1) While the lower MLP sizes greatly reduce Params and MACs with almost negligible accuracy drops ($< -0.5\%$), they lead to relatively less significant improvements in inference latency and train time, compared to the layer effects. 2) Compared to D size, MLP size has better improvements in memory usage, along with similar improvements in other DR metrics. This indicates that MLP size, compared to D size, is more opportunistic in memory-constrained scenarios.

F. SUMMARY ON PATCH AND MODEL SIZE EFFECTS

We summarize important key findings together to build a comprehensive picture of our characterization on patch and model size effects.

Patch Size: When we use a larger patch size on small datasets (e.g., ViT-Ti/32 in our study), more training epochs are required to achieve a certain level of accuracy. However, as long as a target accuracy can be obtained, a larger patch size with a reduced number of tokens (N) can be a better choice for resource-constrained mobile and edge devices, because of the 2.5 - 4.5× higher improvements in terms of important metrics such as MACs, inference latency, train time, GPU memory, energy consumption, and FoM, as shown in the two bottom rows of Tables 15 and 16.

Number of Layers: Among the model size effects, the layer effect is the first-priority factor that needs to be considered in ViT optimization and tuning problems due to the direct improvements in the important metrics (Table 15). However, in layer variation, a problematic optimization issue is that accuracy degradation is more prominent compared to D and M size effects, which are more successful on ViT-Ti/16 (Table 16). The power consumption of ViT models is not directly proportional to the number of layers but is primarily determined by CPU, GPU, and Memory dynamic voltage and frequency scaling (DVFS) governors.

TABLE 17. Results of ablation study on ViT-32 on Laptop and Edge, Batch size = 16 (red, > 2% drop; yellow - acceptable); best within < 2% drop and worst in all). Best values are L:6, D:96, M:192, H:1 and the baseline is: L:12, D:192, M:768, H:3.

P.	Config.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
32	L+D+M+H	-7.9%	87.4%	86.3%	614.7%	21.4%	45.8%	48.1%	60.9%	69.2%	71.0%	12.1%	18.3%	66.5%	193.1%	76.3%	258.8%
		(72.23)	(0.77)	(0.04)	(93.93)	(89.62)	(13)	(2.04)	(565)	(24)	(0.36)	(3482)	(5814)	(5.56)	(3.01)	(1493)	(517.64)
	L+D+M	-4.9%	87.4%	86.3%	643.8%	22.4%	45.8%	48.1%	52.0%	69.2%	70.8%	-11.8%	13.6%	73.3%	205.0%	74.8%	253.1%
		(75.17)	(0.77)	(0.04)	(97.75)	(90.32)	(13)	(2.04)	(693)	(24)	(0.36)	(4429)	(6148)	(5.78)	(3.13)	(1587)	(509.45)
	L+D	-2.9%	75.4%	74.7%	292.1%	15.5%	45.8%	48.8%	35.4%	62.8%	66.4%	-11.5%	-2.2%	77.8%	159.1%	65.7%	153.5%
		(77.15)	(1.50)	(0.07)	(51.54)	(85.22)	(13)	(2.22)	(933)	(29)	(0.41)	(4418)	(7275)	(5.93)	(2.66)	(2160)	(365.68)
	L+M	-4.0%	67.6%	64.8%	192.7%	11.6%	33.3%	48.1%	53.1%	57.7%	65.8%	-8.3%	-12.1%	42.4%	124.4%	61.6%	100.3%
		(76.05)	(1.98)	(0.10)	(38.47)	(82.33)	(16)	(2.04)	(677)	(33)	(0.42)	(4292)	(7975)	(4.75)	(2.30)	(2415)	(288.97)
	D+M	-1.7%	**80.0%**	**77.6%**	**390.0%**	**17.8%**	20.8%	8.1%	**53.1%**	**51.3%**	46.6%	-1.0%	**18.4%**	23.6%	**100.9%**	**56.4%**	**146.0%**
		(78.37)	(1.22)	(0.06)	(64.40)	(86.92)	(19)	(3.62)	(677)	(38)	(0.66)	(4002)	(5810)	(4.12)	(2.06)	(2742)	(354.97)
	L (6)	-0.91%	43.8%	48.2%	76.0%	7.0%	**41.7%**	**42.9%**	33.6%	43.6%	**56.0%**	**13.2%**	-19.1%	**69.5%**	73.5%	47.6%	47.1%
		(79.18)	(3.42)	(0.15)	(23.13)	(79.0)	(14)	(2.25)	(1012)	(44)	(0.54)	(3438)	(8479)	(5.66)	(1.80)	(3300)	(212.24)
	D (96)	**0.13%**	57.1%	57.7%	133.6%	10.1%	12.5%	14.3%	0.0%	28.2%	23.3%	3.26%	14.7%	14.5%	39.5%	34.6%	63.6%
		(80.22)	(2.61)	(0.12)	(30.70)	(81.24)	(21)	(3.38)	(1524)	(56)	(0.94)	(3833)	(6070)	(3.82)	(1.43)	(4116)	(236.00)
	M (192)	-0.05%	45.6%	46.4%	83.7%	7.2%	20.8%	13.2%	33.6%	25.6%	26.9%	10.4%	2.6%	26.2%	34.4%	28.8%	38.1%
		(80.04)	(3.32)	(0.15)	(24.14)	(79.17)	(19)	(3.42)	(1012)	(58)	(0.90)	(3549)	(6929)	(4.21)	(1.38)	(4482)	(199.18)
	default	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.09)	(6.09)	(0.28)	(13.14)	(73.83)	(24)	(3.94)	(1524)	(78)	(1.23)	(3962)	(7117)	(3.34)	(1.03)	(6295)	(144.26)

TABLE 18. Results of ablation study on ViT-16 on Laptop and Edge, Batch size = 32 and 16 (red, > 2% drop; yellow - acceptable); best within < 2% drop and worst in all) Best values are L:8, D:96, M:192, H:2 and the baseline is: L:12, D:192, M:768, H:.

P.	Config.	DA-Sin.			DA-Mul.		DR-Sin.							DR-Mul.			
							Laptop			Edge				Laptop	Edge		
		Acc. ↑	Params (M) ↓	MACs (G) ↓	ID (Acc/ Param) ↑	Net Score ↑	Inf. /bs (ms) ↓	Train time (hours) ↓	GPU Mem. (MB) ↓	Inf. /bs (ms) ↓	Train time (hours) ↓	Mem. (MB) ↓	Power (mW) ↓	AoL (Acc/ Inf.) ↑	AoL (Acc/ Inf.) ↑	Energy /ep. (J) ↓	FoM (Acc/ Pow.×Inf.) ↑
16	L+D+M+H	-2.2%	87.5%	85.0%	676.6%	24.8%	62.3%	78.1%	63.3%	69.0%	70.2%	30.5%	17.6%	157.9%	213.7%	75.5%	280.6%
		(77.92)	(0.71)	(0.19)	(109.6)	(84.38)	(23)	(1.60)	(1461)	(84)	(1.13)	(3566)	(7251)	(3.39)	(0.93)	(5884)	(127.9)
	L+D+M	-1.5%	**87.5%**	**85.0%**	**684.0%**	**25.1%**	**60.7%**	**77.7%**	**63.3%**	**67.2%**	**68.1%**	**26.4%**	17.7%	**149.5%**	**198.9%**	**73.8%**	**263.0%**
		(78.66)	(0.71)	(0.19)	(110.6)	(84.55)	(24)	(1.63)	(1461)	(89)	(1.21)	(3778)	(7243)	(3.28)	(0.88)	(6286)	(122.0)
	L+D	-2.1%	70.7%	71.2%	232.6%	15.2%	44.3%	68.1%	37.9%	59.8%	61.1%	18.9%	14.7%	74.6%	142.0%	66.8%	183.9%
		(78.01)	(1.66)	(0.36)	(46.9)	(77.89)	(34)	(2.33)	(2469)	(109)	(1.47)	(4163)	(7501)	(2.29)	(0.72)	(7948)	(95.4)
	L+M	-0.4%	64.6%	60.8%	181.4%	12.6%	44.3%	69.6%	37.9%	47.2%	49.0%	24.2%	6.8%	78.6%	88.7%	52.5%	102.5%
		(79.79)	(2.01)	(0.49)	(39.7)	(76.11)	(34)	(2.22)	(2469)	(143)	(1.93)	(3888)	(8197)	(2.35)	(0.56)	(11382)	(68.1)
	D+M	-1.0%	82.2%	78.1%	455.4%	20.5%	42.6%	67.4%	37.5%	52.7%	51.8%	16.5%	**20.1%**	72.2%	114.2%	61.4%	167.9%
		(79.17)	(1.01)	(0.28)	(78.4)	(81.48)	(35)	(2.38)	(2485)	(125)	(1.83)	(4284)	(7033)	(2.26)	(0.63)	(9246)	(90.0)
	L (8)	-1.1%	31.3%	32.5%	43.6%	4.2%	31.1%	32.7%	35.9%	30.3%	32.6%	16.0%	2.2%	43.2%	41.4%	34.1%	44.6%
		(79.04)	(3.90)	(0.85)	(20.27)	(77.20)	(42)	(4.92)	(2548)	(189)	(2.55)	(4309)	(8604)	(1.88)	(0.42)	(15808)	(48.61)
	D (96)	-0.62%	57.6%	57.1%	134.2%	10.8%	21.3%	24.3%	0.0%	32.5%	35.3%	9.38%	3.4%	26.1%	46.9%	37.5%	52.0%
		(79.52)	(2.41)	(0.54)	(33.05)	(74.88)	(48)	(5.53)	(3978)	(183)	(2.45)	(4628)	(8503)	(1.66)	(0.43)	(14994)	(51.10)
	M (192)	-0.2%	48.9%	43.2%	95.2%	7.9%	24.6%	27.0%	36.8%	23.2%	19.4%	17.8%	5.6%	32.3%	31.9%	28.6%	39.8%
		(79.95)	(2.90)	(0.71)	(27.55)	(72.95)	(46)	(5.33)	(2468)	(205)	(2.9)	(4219)	(8303)	(1.74)	(0.39)	(17111)	(46.97)
	default	**0.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		(80.14)	(5.68)	(1.26)	(14.1)	(67.61)	(61)	(7.30)	(3978)	(264)	(3.78)	(5133)	(8798)	(1.31)	(0.30)	(23975)	(33.6)

D Size: The D size effect yields the highest improvements across all the device-agnostic (DA) metrics, without accuracy drop in ViT-Ti/32 and only a negligible 0.62% drop in ViT-Ti/16 (Tables 15 and 16). However, the best improvements in DA metrics do not guarantee the highest improvements in DR metrics (i.e., mostly the best improvements of DR metrics are observed in the layer effects (L), with the highlighted (bold) values especially in Table 15). Moreover, compared to the number of layers and MLP size, D size variations have no or little effect on the memory usage.

MLP Size: The MLP size effect provides significant improvements in DA metrics with almost negligible accuracy drops (less than 0.5%), although they are not the best improvements. However, in terms of memory usage, it has better improvements compared to the D size and layer effects; therefore, the MLP size optimization and tuning can be more opportunistic for memory-constrained devices.

Number of Heads: The head size effect has the least impacts and improvements on overall model performance, along with the observation that ViT-Ti/16 experiences more serious trade-offs between accuracy and the number of heads.

Lastly, we summarize in terms of the number of best improvements in each effect out of all the 16 measurements: In ViT-Ti/32, the top layer effects (9/16), followed by D size (7/16), H size (0/16) and MLP size (1/16, the same with L) (Table 15). In ViT-Ti/16, the order looks a bit different: The top D size (9/16), followed by the number of layers (3/16), Head size (0/16) and MLP size (3/16) (Table 16). In short, the layer (L) effect dominates in ViT-Ti/32, while the D size effect dominates in ViT-Ti/16.

G. ABLATION STUDY

We present the results of the ablation study using the combinations of the local optima (i.e., the number of layers of 6, D size of 96, and MLP size of 192 in ViT-Ti/32 and the number of layers of 8 and the same D and MLP sizes in ViT-Ti/16 in Tables 17 and 18).

1) ABLATION RESULTS OF ViT-Ti/32.

As shown in Table 17, when we apply the 2% accuracy degradation constraint, the D+M combination yields the highest improvements in terms of most metrics except accuracy and a few DR metrics. Moreover, when the local optimal L (6 layers) is combined with another optimal or optima (i.e., L+M, L+D, L+D+M, and L+D+M+H), there are clear trade-offs between accuracy and other metrics (especially time-related metrics).

Optmization Guidelines: Through the ablation study, we give the following guidelines for model size optimizations/tuning: 1) Firstly, for a design goal of *significant improvements without accuracy degradation*, D size or M size optimization/tuning is most recommended. 2) Secondly, for a design goal of *maximum improvements with minimal accuracy degradation (within 2%)*, L or D+M size optimization is more recommended. 3) Lastly, sacrificing more accuracy, L+local optimal/optima configurations could be used for minimizing MACs, inference latency, and train time etc.

2) ABLATION RESULTS OF ViT-Ti/16, COMPARED TO ViT-Ti/32

As shown in Table 18, the main difference compared to ViT-Ti/32 is that there are the three L included combinations (i.e., L+M, L+D+M and D+M) without direct trade-offs between accuracy and time-related metrics. We speculate that it happens because the local optimal of L is not 6 layers but 8 layers in ViT-Ti/16.

Optmization Guidelines: We give the following guidelines for model size optimizations/tuning: 1) Firstly, for a design goal of *significant improvements without accuracy degradation*, optimizing or tuning the D size or MLP size is most recommended. 2) Secondly, for a design goal of *maximum improvements with minimal accuracy degradation (within 2%)*, it is recommended to consider L, D+M, or a combination that includes L.

VI. DISCUSSION AND FUTURE WORK

Although we comprehensively investigate the effects of input patch size and model size on competitive efficient-ViTs, a potential limitation of our work is the evaluation using only the CIFAR-10 dataset, which may affect the generalizability to other datasets. However, in this work, we focus more on optimization factors and our HelLo methodology (i.e., the hierarchical and local search approach to quickly search local-optimal efficient-ViTs within k% accuracy degradation), which can be applicable for other datasets as well. When we consider another medical dataset (COVID-19 infected lungs) with a higher resolution (512 × 512) [58], candidates of baseline models and patch sizes can be selected according to the k% accuracy degradation scheme by evaluating accuracies of baseline ViT models, compared to the accuracy of the baseline ViT of the work (i.e., 94.03%).

The limitations of this study will be addressed in our future work: First, we will expand our HelLo method to medium-sized datasets to address the scalability issue, as certain mobile and edge devices may utilize high-resolution datasets in some domains. Second, we will conduct more challenging investigations into device-related (DR) differences, including: 1) the characteristics between inference and training phases and 2) the characterization of GPU parallelism and memory parallelism on specific frameworks using profiling (e.g., *nvprof*) tools. Lastly, we will focus more on on-device training optimization for efficient-ViTs based on our key findings and previously studied on-device training on CNNs [59], [60], fine-tuning in transfer learning (TL) [61], and ViTs for federated learning (FL) [62].

VII. CONCLUSION

In this paper, we conduct a comprehensive patch and model size characterization and optimization for efficient ViT models on small datasets using quantitative twelve (five DA + seven DR) metrics, improving performances on resource-constrained mobile and edge AI devices. We thoroughly investigate and summarize key findings of the effects of various hyper-parameters and the ablation study, considering input patch sizes (P) and model sizes of number of layers (L), D size (D), number of heads (H), and MLP size (M).

The key findings are summarized as follows: A large patch size reduces computational overhead and improves overall latency, memory efficiency, and energy efficiency, as long as the target accuracy can be achieved with the increased number of epochs. In terms of model size optimization, it depends on design goals: D size or MLP size (or less aggressive L) optimizations are recommended for a design goal of *significant improvements without accuracy degradation*, whereas L or D+M (or less aggressive L + M and/or D) is recommended for a design goal of *maximum improvements with minimal accuracy degradation*.

With our hierarchical and local optimization/tuning (HelLo) method for efficient ViTs, we achieve significant improvements up to 85% in MACs, 67.2% in inference latency, 77.7% in train latency/time, 63.3% in GPU memory, 73.8% in energy consumption, and 263.0% in FoM, within minimal accuracy degradation (up to 2%).

REFERENCES

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–22.

[2] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers & distillation through attention," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, vol. 139, M. Meila and T. Zhang, Eds. PMLR, Jul. 2021, pp. 10347–10357.

[3] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10012–10022.

[4] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 568–578.

[5] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12104–12113.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[8] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 843–852.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K. Berlin, Germany: Springer-Verlag, 2020, pp. 213–229.

[10] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA: Curran Associates, 2024, pp. 1–14.

[11] L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, and Y. Qiao, "VideoMAE v2: Scaling video masked autoencoders with dual masking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 14549–14560.

[12] J. Pan, A. Bulat, F. Tan, X. Zhu, L. Dudziak, H. Li, G. Tzimiropoulos, and B. Martinez, "EdgeViTs: Competing light-weight CNNs on mobile devices with vision transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 294–311.

[13] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. H. Tay, J. Feng, and S. Yan, "Tokens-to-token ViT: Training vision transformers from scratch on ImageNet," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 558–567.

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[15] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[16] X. Wang, L. L. Zhang, Y. Wang, and M. Yang, "Towards efficient vision transformer inference: A first study of transformers on mobile devices," in *Proc. 23rd Annu. Int. Workshop Mobile Comput. Syst. Appl.*, NY, NY, USA, Mar. 2022, pp. 1–7.

[17] M. Maaz, A. Shaker, H. Cholakkal, S. Khan, S. W. Zamir, R. M. Anwer, and F. Shahbaz Khan, "EdgeNeXt: Efficiently amalgamated CNN-transformer architecture for mobile vision applications," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer-Verlag, 2022, pp. 3–20.

[18] Y. Li, G. Yuan, Y. Wen, J. Hu, G. Evangelidis, S. Tulyakov, Y. Wang, and J. Ren, "EfficientFormer: Vision transformers at MobileNet speed," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 12934–12949.

[19] Y. Li, J. Hu, Y. Wen, G. Evangelidis, K. Salahi, Y. Wang, S. Tulyakov, and J. Ren, "Rethinking vision transformers for mobilenet size and speed," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2023, pp. 16889–16900.

[20] H. Cai, J. Li, M. Hu, C. Gan, and S. Han, "EfficientViT: Lightweight multi-scale attention for high-resolution dense prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2023, pp. 17302–17313.

[21] J. Castrillo, R. Valle, and L. Baumela, "Efficiency evaluation of mobile vision transformers," in *Proc. Int. Conf. Inf. Technol. Syst.* Cham, Switzerland: Springer, 2024, pp. 3–11.

[22] S. Lee, S. Lee, and B. C. Song, "Improving vision transformers to learn small-size dataset from scratch," *IEEE Access*, vol. 10, pp. 123212–123224, 2022.

[23] Y. Liu, E. Sangineto, W. Bi, N. Sebe, B. Lepri, and M. Nadai, "Efficient training of visual transformers with small datasets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 23818–23830.

[24] H. Gani, M. Naseer, and M. Yaqub, "How to train vision transformer on small-scale datasets?" in *Proc. 33rd Brit. Mach. Vis. Conf. (BMVC)*. London, U.K.: BMVA Press, 2022, pp. 1–19.

[25] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the big data paradigm with compact transformers," 2021, *arXiv:2104.05704*.

[26] J. Hong Tan, "Pre-training of lightweight vision transformers on small datasets with minimally scaled images," 2024, *arXiv:2402.03752*.

[27] D. Shan and G. chen, "GvT: A graph-based vision transformer with talking-heads utilizing sparsity, trained from scratch on small datasets," 2024, *arXiv:2404.04924*.

[28] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "DPP-Net: Device-aware progressive search for pareto-optimal neural architectures," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 517–531.

[29] C.-H. Hsu, S.-H. Chang, J.-H. Liang, H.-P. Chou, C.-H. Liu, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, and D.-C. Juan, "MONAS: Multi-objective neural architecture search using reinforcement learning," 2018, *arXiv:1806.10332*.

[30] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.

[31] C. White, S. Nolen, and Y. Savani, "Exploring the loss landscape in neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2021, pp. 654–664.

[32] T. Den Ottelander, A. Dushatskiy, M. Virgolin, and P. A. Bosman, "Local search is a remarkably strong baseline for neural architecture search," in *Proc. 11th Int. Conf. Evol. Multi-Criterion Optim. (EMO)*, Shenzhen, China. Cham, Switzerland: Springer, 2021, pp. 465–479.

[33] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "GLiT: Neural architecture search for global and local image transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12–21.

[34] NVIDIA. (2020). *Tegrastats Utility*. Accessed: Jun. 20, 2023. [Online]. Available: https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html

[35] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implement. (OSDI)*.Savannah, GA, USA: USENIX Association, 2016, pp. 265–283.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–12.

[37] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[38] Y. Wang, R. Huang, S. Song, Z. Huang, and G. Huang, "Not all images are worth 16×16 words: Dynamic transformers for efficient image recognition," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2024, pp. 1–14.

[39] L. Beyer, P. Izmailov, A. Kolesnikov, M. Caron, S. Kornblith, X. Zhai, M. Minderer, M. Tschannen, I. Alabdulmohsin, and F. Pavetic, "FlexiViT: One model for all patch sizes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 14496–14506.

[40] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, *arXiv:1605.07678*.

[41] A. Wong, "NetScore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage," 2018, *arXiv:1806.05512*.

[42] M. Mohammadi, H. Smith, L. Khan, and R. Zand, "Facial expression recognition at the edge: CPU vs GPU vs VPU vs TPU," in *Proc. Great Lakes Symp. VLSI*, Jun. 2023, pp. 243–248.

[43] D. Velasco-Montero, J. Fernández-Berni, R. Carmona-Galán, and Á. Rodríguez-Vázquez, "Performance analysis of real-time DNN inference on Raspberry Pi," in *Real-Time Image and Video Processing Conference, SPIE Defense + Commercial Sensing Symposium*. OR, USA: The International Society for Optics and Photonics, 2018.

[44] H. Zhang, W. Hu, and X. Wang, "Fcaformer: Forward cross attention in hybrid vision transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2023, pp. 6060–6069.

[45] Y.-L. Liao, S. Karaman, and V. Sze. *ViT-Search: Compute_Flop_MAC*. Accessed: Jun. 20, 2023. [Online]. Available: https://github.com/yilunliao/vit-search/blob/master/network_utils/compute_flop_mac.py

[46] Y.-L. Liao, S. Karaman, and V. Sze, "Searching for efficient multi-stage vision transformers," 2021, *arXiv:2109.00642*.

[47] K. Yoshida, R. Sageyama, S. Miwa, H. Yamaki, and H. Honda, "Analyzing performance and power-efficiency variations among NVIDIA GPUs," in *Proc. 51st Int. Conf. Parallel Process.*, New York, NY, USA, 2023, pp. 1–12.

[48] M. Dehghani, B. Mustafa, J. Djolonga, J. Heek, M. Minderer, M. Caron, A. Steiner, J. Puigcerver, R. Geirhos, I. M. Alabdulmohsin, A. Oliver, P. Padlewski, A. Gritsenko, M. Lucic, and N. Houlsby, "Patch n'pack: NaViT, a vision transformer for any aspect ratio and resolution," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–23.
[49] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Commun. ACM*, vol. 64, no. 3, pp. 107–115, Feb. 2021.
[50] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018.
[51] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6023–6032.
[52] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Apr. 2020, pp. 13001–13008.
[53] R. Wightman. *Timm: PyTorch Image Models*. Github Repository. Accessed: Jun. 20, 2023. [Online]. Available: https://github.com/rwightman/pytorch-image-models
[54] M. Karzhaubayeva, A. Amangeldi, and J.-G. Park, "CNN workloads characterization and integrated CPU–GPU DVFS governors on embedded systems," *IEEE Embedded Syst. Lett.*, vol. 15, no. 4, pp. 202–205, Dec. 2023.
[55] NVIDIA Corporation. *NVIDIA Visual Profiler (Nvprof)*. Accessed: Jun. 20, 2023. [Online]. Available: https://docs.nvidia.com/cuda/profiler-users-guide/index.html
[56] S. Nag, G. Datta, S. Kundu, N. Chandrachoodan, and P. A. Beerel, "ViTA: A vision transformer inference accelerator for edge applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5.
[57] J. Tao, Z. Gao, and Z. Guo, "Training vision transformers in federated learning with limited edge-device resources," *Electronics*, vol. 11, no. 17, p. 2638, Aug. 2022.
[58] J. C. M. Than, P. L. Thon, O. M. Rijal, R. M. Kassim, A. Yunus, N. M. Noor, and P. Then, "Preliminary study on patch sizes in vision transformers (ViT) for COVID-19 and diseased lungs classification," in *Proc. IEEE Nat. Biomed. Eng. Conf. (NBEC)*, Nov. 2021, pp. 146–150.
[59] Y. Wang, Z. Jiang, X. Chen, P. Xu, Y. Zhao, Y. Lin, and Z. Wang, "E2-train: Training state-of-the-art CNNs with over 80% energy savings," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.
[60] Y. Wu, Z. Wang, Y. Shi, and J. Hu, "Enabling on-device CNN training by self-supervised instance filtering and error map pruning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3445–3457, Nov. 2020.
[61] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce activations, not trainable parameters for efficient on-device learning," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020, pp. 1–14.
[62] X. Zuo, Y. Luopan, R. Han, Q. Zhang, C. H. Liu, G. Wang, and L. Y. Chen, "FedViT: Federated continual learning of vision transformer at edge," *Future Gener. Comput. Syst.*, vol. 154, pp. 1–15, May 2024.

JURN-GYU PARK (Associate Member, IEEE) received the B.S. degree in mechanic and automotive engineering from Kookmin University, Seoul, South Korea, in 2001, the M.S. degree in computer engineering from Yonsei University, Seoul, in 2012, and the Ph.D. degree from the University of California at Irvine, USA, in 2017. From 2002 to 2011, he was a System Software Engineer and a Technical Lecturer with MDS Technology, Seoul. Since 2018, he has been an Assistant Professor with the Computer Science Department, Nazarbayev University, Astana, Kazakhstan. His research interests include DL (especially CNNs and ViTs) performance optimization on edge-AI and embedded-AI devices, machine learning (ML) enhanced prediction model building methodologies, and energy-efficient CPU-GPU dynamic power management (DPM) methodologies for mobile HMPSoCs.

AIDAR AMANGELDI received the bachelor's degree in computer science from Nazarbayev University, Astana, Kazakhstan, in 2024, where he is currently pursuing the master's degree in data science. Since August 2022, he has been a Research Assistant with Nazarbayev University, focusing on characterizing vision transformer and CNN workloads on embedded systems and developing CPU-GPU integrated DVFS governor algorithms. His research interests include machine learning, embedded systems, and computer vision.

NAIL FAKHRUTDINOV is currently pursuing the bachelor's degree in computer science with Nazarbayev University, Astana, Kazakhstan. Since 2021, he has been a Research Assistant with the School of Engineering and Digital Sciences. His current research interests include generative AI, brain–computer interfaces, and computer vision.

MERUYERT KARZHAUBAYEVA received the B.S. degree in computer science from Nazarbayev University, Astana, Kazakhstan, in 2023. She is currently pursuing the M.S. degree in information security policy and management with the Heinz College, Carnegie Mellon University, Pittsburgh, PA, USA. She has nearly a year of experience in security services with PwC. She was a Research Assistant with Nazarbayev University, for two years, focusing on CPU-GPU utilization optimization under CNN workloads on embedded devices. Her research interests include embedded systems, security implications of the Internet of Things, and threat intelligence.

DIMITRIOS ZORBAS (Member, IEEE) received the Ph.D. degree in computer science from the University of Piraeus, Greece. He is an Assistant Professor with Nazarbayev University, Kazakhstan. He was a Postdoctoral Researcher with Inria Lille–Nord Europe, and the University of La Rochelle, France. He was a Researcher with the Tyndall National Institute, University College Cork, Ireland, after receiving a Marie Curie Fellowship. He has worked in several national and FP7 and H2020 projects. He is the author of more than 80 peer-reviewed publications in the area of wireless communications, protocols, and edge computing.