Tiny CNNs or Tiny Transformers? A Systematic Analysis of Latency, Accuracy and Domain Shift Robustness in Extreme-Edge Image Classification

Călin Diaconu*, Luca Bompani*, Alberto Dequino*, Davide Nadalini†, Francesco Conti*
*DEI, University of Bologna, Bologna, Italy
†PSI, Katholieke Universiteit Leuven, Leuven, Belgium

***Abstract*—Vision Transformers have recently surpassed convolutional networks on several benchmark tasks, but their benefits on extreme-edge IoT hardware remain unclear due to higher computational and memory demands of attention-based blocks. This paper provides a systematic comparison of tiny CNN, Transformer, and hybrid architectures for image classification under resource constraints. We evaluate models using hardware-agnostic metrics (FLOPs, parameter count, and accuracy) and hardware-specific measurements (working-memory footprint and on-device inference latency) on an 8-core RISC-V microcontroller with a 2 MiB on-chip memory budget. Accuracy is assessed in-domain and under two out-of-distribution transfer settings: a dataset shift at fixed resolution and a combined dataset-and-resolution shift. Under hardware-agnostic profiling, attention-based models achieve higher accuracy than CNNs at the same input resolution (up to +3.81% in-domain and +6.67% under dataset shift) while requiring up to 10× more operations. When the input resolution is reduced, this advantage is not preserved and CNNs outperform by up to 16.4%. On-device measurements show that, among models that fit within the memory budget, the best attention-based model (CaiT) provides 21% higher accuracy than the fastest CNN, at a 7× latency cost, while also exhibiting a more uniform layer-wise working-memory profile. Overall, the results indicate that selecting tiny models for edge IoT deployment requires hardware-aware profiling that jointly accounts for accuracy under distribution and resolution changes, working-memory behavior, and latency, with CNNs remaining competitive when latency is the primary constraint.**

I. Introduction

In recent years, Transformer models [38] have revolutionized the field of Artificial Intelligence (AI), consistently outperforming Convolutional Neural Networks (CNNs) in vision tasks [13] and Recurrent Neural Networks (RNNs) in natural language processing [5]. However, despite their success, Transformers require substantial computational resources and memory due to the large matrix multiplications and Softmax operations performed in their foundational blocks, i.e., the attention layers (see Fig. 1 for a comparison with CNNs). These issues make their deployment challenging on resource-constrained embedded platforms, such as MicroControllers (MCUs) and Ultra-Low-Power (ULP) Systems-on-Chip (SoCs), which are widely used in edge Internet-of-Things (IoT) applications.

Nevertheless, recent work has demonstrated the feasibility of deploying tiny Transformers on extreme-edge devices [42], [4], [8], such as MCUs, where CNNs still offer a state-of-the-art trade-off between latency, memory usage, and accuracy.

Fig. 1: Maximum working memory vs. number of floating point operations of models listed in Table I and larger variants belonging to the same model families, evaluated on ImageNet, on a 224×224 input.

Model	Year	Params	FLOPs	Top-1 Acc	Type
TinyViT-5M [42]	2022	5.4M	1.3G	79.1–80.7%	Hybrid
CaiT-XXS-24 [36]	2021	12M	2.5G	78.4%	Attention
FastViT-T8 [37]	2023	4M	0.7G	77.2%	Hybrid
EfficientNet-B0 [34]	2019	5.3M	0.39G	77.1%	CNN
EfficientFormerV2-S0 [20]	2023	3.5M	0.4G	76.2%	Hybrid
DeiT-Tiny [35]	2020	5.7M	1.26G	72.2–74.5%	Attention
EdgeNeXt-XXS [24]	2022	1.3M	0.2G	71.2%	Hybrid
PVT-v2-B0 [39]	2022	3.7M	0.6G	∼70%	Hybrid
MobileViTv2-0.5x [26]	2022	∼1.5M	0.5G	∼70%	Hybrid
MobileViT-XXS [25]	2021	1.3M	0.4G	69.0%	Hybrid
MobileNetV3 Small 0.75x [14]	2019	2.4M	44M	65%	CNN
ShuffleNetV2 0.5x [23]	2018	1.4M	41M	61.1%	CNN
MobileNetV2 0.35x [32]	2018	1.7M	66M	60.3%	CNN
SqueezeNet V1.1 [16]	2016	1.2M	0.35G	58.2%	CNN
ShuffleNetV1 0.5x [43]	2018	0.7M	38M	57.3%	CNN
MobileNetV1 0.25x [15]	2017	0.5M	41M	50.0%	CNN

TABLE I: List of tiny vision models below 12M parameters. The smallest model of each model family is included. The top-1 accuracy is reported on ImageNet [31].

However, existing works overlook a systematic analysis of the trade-off between the additional computational and memory costs introduced by tiny Transformers relative to CNNs and their prediction accuracy, particularly under exposure to unseen data distributions (i.e., domain shifts). This phenomenon has been shown to critically degrade the accuracy of tiny CNNs [27]. It has motivated the development of novel On-Device (Continual) Learning algorithms to enable model self-adaptation at the edge [29].

Therefore, at the current state-of-the-art, a key question remains unanswered: *what practical benefits do tiny Transform-*

ers provide over CNNs on severely resource-limited hardware? Focusing on image classification, this work presents a comprehensive study of tiny AI models for edge IoT deployment, examining the trade-offs between CNN- and Transformer-based architectures. We compare these models from a joint perspective of computational complexity (floating-point operations per inference), parameter count, and classification accuracy, factors that determine deployability in hardware-agnostic yet resource-constrained settings. We then extend the analysis to a hardware-specific scenario, introducing two additional evaluation metrics: *working memory* and *inference latency*. To the best of our knowledge, such a holistic evaluation has not been reported previously in this application domain, and it is essential for informed model selection in real-world IoT systems. To investigate the critical aspect of classification accuracy in such tiny models, we evaluate model performance under both in-domain and out-of-distribution settings. First, we consider an in-domain scenario in which the training and test data are drawn from the same distribution. Second, we consider two transfer learning settings with a frozen backbone and a trained classification head: (i) *domain shift only*, where we train our detection head and evaluate on a different dataset at the exact input resolution as pre-training; and (ii) *domain + resolution shift*, where we evaluate on the same dataset of the previous step while changing the input resolution relative to pre-training.

Our results show that tiny Transformers achieve higher accuracy than CNNs in both in-domain and out-of-distribution settings, but at a steep efficiency cost under a hardware-agnostic profiling lens. In the $\approx$1M-parameter regime, the best Transformer improves over the strongest CNN by +3.81% in-domain and +6.67% for out-of-distribution, while requiring roughly $10\times$ more FLOPs at the same input size. Moreover, this accuracy margin is not robust to reduced input resolution, which is common in low-power IoT devices, either due to sensor constraints or a deliberate choice to reduce energy consumption via downsampling. In this setting, the Transformer accuracy advantage disappears, and it instead falls 16.41% behind CNNs. Overall, these results indicate that state-of-the-art tiny Transformers do not offer a consistent efficiency advantage on edge IoT devices; in practice, their adoption depends on a careful balance of accuracy, memory footprint, and compute, and CNNs remain a strong baseline for tiny IoT deployments.

When considering deployment on an 8-core RISC-V microcontroller [30], hardware-specific effects become decisive. Among models fitting within the 2 MiB on-chip memory budget, Transformer architectures exhibit more uniform layer-wise memory profiles, with no layers exceeding the memory threshold compared to 11.4% for CNNs. The best attention-based model (CaiT-XXS-24) achieves 21% higher accuracy than the fastest CNN at 9.41 billion cycles, while CNNs retain a $1.5\times$ latency advantage, with the fastest CNN requiring only 1.37 billion cycles. These results show that while CNNs remain the best choice when overall latency is the primary objective, Transformers provide higher accuracy and more bounded layer-wise memory requirements, which improves deployability under tight on-chip budgets and indicates strong potential for use in constrained edge environments.

II. Related Work

A. Transformer Deployment on Extreme Edge Devices

Recent studies have investigated how to adapt Transformer architectures to the stringent memory and computational budgets of IoT hardware. MCUFormer [21], for instance, leverages Neural Architecture Search (NAS) [10] to generate Vision Transformer (ViT) variants tailored for microcontroller units (MCUs). Although the approach specifically targets STM32 devices, it provides only limited analysis of memory footprint, end-to-end latency, and performance beyond ImageNet, leaving open questions regarding real-world deployability and generalization.

Burrello et al. [4] introduce a dedicated runtime stack with optimized kernels to reduce data marshalling overhead when executing hybrid CNN-Transformer models on MCUs. Their framework combines FP32 and INT8 quantization techniques (e.g., I-BERT [17] and NEMO [7]) to lower memory usage and latency. However, the evaluation is confined to a single application domain using the TinyRadarNN dataset [33], providing limited insight into how well these methods generalize across different vision tasks or hardware platforms.

MobileViT [25] proposes a lightweight architecture that integrates convolutional layers with Transformer blocks at a scale comparable to MobileNet [14]. While it improves ImageNet-1K accuracy and generalization by approximately 2% over MobileNet, its inference latency can be up to $9\times$ higher at similar parameter counts, raising concerns about its suitability for latency-sensitive edge deployments.

In contrast to these efforts, we focus on benchmarking state-of-the-art tiny Transformer models alongside CNNs to systematically analyze the trade-offs involved in deploying these architectures on extreme-edge devices.

B. Comparison Between CNNs and Transformers

Several studies have compared CNNs and Transformers on vision tasks from an architectural and accuracy perspective. Bai et al. [1], for example, evaluate ResNet-50 [18] and DeiT [35] on ImageNet [31], ensuring a fair comparison in terms of model size and training strategy. Their results show that CNNs are equally or more robust to specific perturbation- and patch-based adversarial attacks. In contrast, Transformers achieve superior performance on out-of-distribution data, with accuracy gains ranging from 4.7% to 10% across benchmarks. However, their analysis is restricted to large-scale models and does not account for memory footprint, FLOPs, or latency, factors that are critical for deployment on low-power hardware.

Similarly, Wang et al. [40] introduce ViT-inspired design elements into a ResNet architecture, such as patchified stems [9] and larger convolutional kernels. Their modified model achieves up to a 5.8% improvement in accuracy over DeiT-B on out-of-distribution data. Nevertheless, this performance gain comes at a computational cost comparable to DeiT,

and the study does not include a hardware-aware evaluation to assess the trade-off between accuracy and efficiency in resource-constrained environments.

In contrast to these works, we present a systematic analysis of CNNs and Transformers specifically designed for extreme-edge devices. Beyond model size and accuracy, we evaluate on-device latency, memory usage, and robustness to domain shift, enabling a more comprehensive assessment of which architectures are best suited for deployment under stringent hardware constraints.

C. Domain Shift in Tiny Edge AI

Memory and computational constraints of extreme-edge devices limit the deployment of AI in IoT systems to very small, task-specific models, which often exhibit limited generalization capability.

Prior works, such as Lotti et al. [22], indicate that large Transformer architectures may adapt more effectively than similarly sized CNNs in specialized applications. However, to the best of our knowledge, no study has examined whether this advantage persists under the tight resource budgets and application constraints typical of IoT and extreme-edge scenarios.

Domain shift in IoT contexts has been predominantly investigated in CNN-based models, with studies highlighting their pronounced performance degradation when exposed to out-of-distribution data. For example, in monocular depth estimation, Nadalini et al. [27] show that a 107k-parameter CNN achieves less than 20% accuracy when evaluated on a different outdoor setting from the KITTI dataset [12].

Building on these findings, we present a systematic study of domain shift in image classification for extreme-edge devices. We compare more than 10 tiny models, including CNNs, Transformers, and hybrid architectures, under identical domain-shift scenarios, providing the first comprehensive cross-family evaluation of model robustness at the extreme edge.

III. Method

We assess the deployability of CNNs and Transformer-based models, and their performance on edge platforms, using a set of complementary metrics that jointly form a hardware-agnostic benchmarking suite for edge AI deployment.

Specifically, we evaluate the following: *Number of parameters (in millions)* is a proxy for representational capacity and storage footprint; when multiplied by the number of bytes used to store weights, it must fit within the available on-board non-volatile memory (e.g., flash). *FLOPs per inference (in billions)* measures the computational complexity of a single inference, accounting for dominant operations such as matrix multiplications, non-linear activations, and Softmax, and is commonly used as an indicator of inference latency and energy consumption on the target hardware. *Classification accuracy* (percentage of correct predictions) is the test accuracy on data drawn from the same distribution as the training set, representing upper-bound performance under nominal conditions. *Robustness to domain shift* (percentage of correct predictions) is the test accuracy on data from unseen distributions, capturing performance degradation under domain shift and indicating model reliability and the potential need for retraining or adaptation during deployment.

When selecting a model for deployment on embedded hardware, these hardware-agnostic metrics are complemented with two hardware-aware metrics. *Working memory footprint (in MB)* is the peak memory required during inference and must fit within the available on-chip RAM, cache, and/or scratchpad; larger footprints can increase latency and energy due to tiling and additional data transfers. *Inference latency (in clock cycles)* is measured directly on the target platform using on-board performance counters.

For the specific workload (image classification in our case), the best-suited model is selected as the one that achieves the best trade-off on the target hardware among those selected during the hardware-agnostic evaluation. We provide an example of our study in Sec. V.

IV. Experimental Setup

A. Target Models

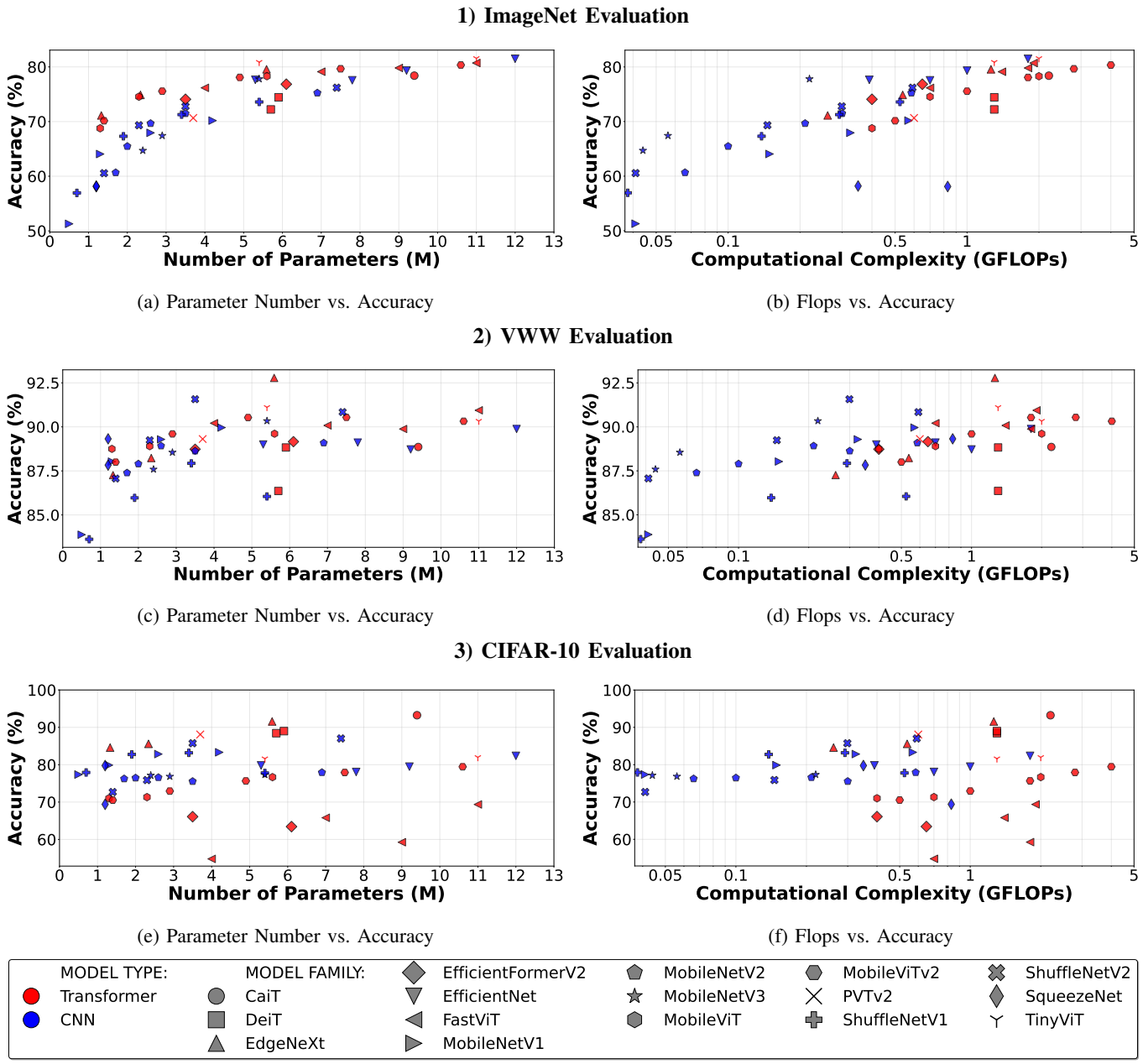
In line with the considerations in Sec. III, we select CNN- and Transformer-based architectures with a maximum parameter count of 12 million. The target models included in our profiling study are listed in Table I, which reports their ImageNet Top-1 accuracy (as stated in the corresponding reference papers and in PyTorch Image Models (Timm) [41]) together with their computational complexity. We verified all reported complexity figures using a 224×224 RGB input for every model. Finally, we group the selected architectures into three categories: (i) pure CNNs, (ii) pure attention-based transformers, and (iii) hybrid attention-CNN designs.

B. Datasets & Benchmark

We evaluate on two standard TinyPerf benchmarks [2]: CIFAR-10 [19] for general-purpose object classification and Visual Wake Words (VWW) [6] for application-oriented visual recognition. For both datasets, we hold out 10% of the original training set for validation, resulting in 45000/5000/10000 train/val/test images for CIFAR-10 and 103500/11500/5000 samples for VWW. These splits are kept fixed across all experiments, including those at different input resolutions. In particular, for CIFAR-10, we consider the native 32×32 resolution as well as an upscaled 224×224 version obtained via bilinear interpolation. In contrast, for VWW, we consider 96×96 and 224×224, consistent with the range of input sizes discussed in the original benchmark.

C. Training and Evaluation Setup

All training and evaluation runs are performed on A100 GPUs using PyTorch 2.0.0 and CUDA 12.1. Our protocol starts from ImageNet-pretrained networks and comprises three complementary evaluations. We first quantify in-distribution performance by evaluating each pretrained model on the ImageNet validation set at 224×224. We then study domain



(a) Parameter Number vs. Accuracy

(b) Flops vs. Accuracy

(c) Parameter Number vs. Accuracy

(d) Flops vs. Accuracy

(e) Parameter Number vs. Accuracy

(f) Flops vs. Accuracy

Fig. 2: Accuracy vs. number of parameters and floating point operations on the ImageNet, VWW and CIFAR-10 Datasets. For all evaluations we consider a 224×224 input image.

transfer by adapting the pretrained backbones to each target dataset via head-only fine-tuning, initially keeping the ImageNet input resolution (224×224) to isolate the effect of the dataset shift.

Finally, to characterize the impact of input size, we repeat the same head-only adaptation at the dataset-native resolution, namely 32×32 for CIFAR-10 and 96×96 for VWW, and evaluate using the corresponding fixed splits.

All training in our evaluation uses the same optimizers and hyperparameters as in the original papers presenting each model, with only two exceptions.

Mobilenetv3 [14] and EfficientNet [34] have been trained using an AdamW optimizer with a learning rate set to $5 * 10^{-4}$ and weight decay set to $5 * 10^{-2}$, as the original works' configuration failed to converge in our testing framework. All trainings use the same augmentations: a random Horizontal flip and a random crop, both with an execution probability of 50%.

To evaluate working memory, we compute the worst-case per-layer memory required by a given model. This offers a finer-grained profiling compared to an end-to-end approach, as it enables precise identification of the layers which consume disproportionate memory or compute power. It is equal to the

maximum memory occupied by any single layer of that model, including the inputs, the required buffers (including the output buffer), and the weights and biases. We omit auxiliary code elements, such as kernel-required loop index variables, as they constitute a negligible portion of the memory needed and are implementation-dependent. The network's FLOPs are instead obtained using Facebook Research's fvcore library [11].

D. Deployment Flow

For deployment, we use PULP-TrainLib [28] as an optimized DNN inference library for multi-core RISC-V MCUs, enabling efficient on-device execution on ultra-low-power platforms. The library provides a range of performance-tunable DNN layer primitives implemented via matrix-multiplication-based kernels, along with common activation and support functions required for end-to-end inference.

To optimize kernel performance, PULP-TrainLib is coupled with a hardware-in-the-loop method that greedily selects, layer by layer, the primitive that minimizes latency given the network structure and the memory constraints of the target device. Results show that this approach improves MAC/clk by up to 2.4× over a matrix-multiplication baseline, reaching 4.39 MAC/clk.

PULP-TrainLib requires PULP-SDK [3] and the RISC-V GNU GCC Toolchain to be used and compiled. The library targets a PULP SoC instance that includes an MCU for control-related tasks and a multi-core cluster for parallel computation, where the cluster includes 8 RISC-V cores. Each core implements a basic set of standard RISC-V extensions (RV32IMFC), and a custom extension (Xpulp) provides DSP-like features including post-increment load/store operations and 2-level nested hardware loops.

We evaluate on all 8 cores available in a PULP-Cluster architecture [30], layer-by-layer, in L2, which is limited at 2 MiB. The obtained values are then summed into the total per-model number of cycles. Because of the memory constraint, we only evaluate the models with a working memory below 2 MiB. This leads us to a list comprising 2 transformer models, 2 hybrid ones, and 3 CNNs.

V. Results

A. Accuracy vs. Parameters vs. Flops Tradeoffs

Fig. 2 reports the results of our experimental setup on ImageNet and on the two selected image classification datasets, CIFAR-10 and Visual Wake Words (VWW). The corresponding numerical results are provided in Table II. In the figures, pure CNN models are highlighted in blue, while architectures incorporating attention mechanisms are shown in red. For this evaluation, we do not distinguish between pure transformers and hybrid models for visual clarity. Different marker shapes identify the model families. Multiple points with the same shape correspond to the same architectural design evaluated at different model sizes. For clarity, full model names and sizes are omitted from the plots.

	ImageNet 224×224	VWW 224×224	CIFAR-10 224×224	VWW 96×96	CIFAR-10 32×32
MobileNet v1 0.25x	60.66	87.39	76.24	81.25	60.84
MobileNet v2 0.5x	51.29	83.88	77.36	78.62	**65.16**
MobileNet v3 Small 0.75	64.7	87.59	77.14	72.3	35.94
DeiT-Tiny	74.44	88.83	88.99	72.39	42.35
CaiT-XXS-24	78.4	88.86	**93.24**	N/A	N/A
EdgeNeXt-XXS	71.12	87.26	84.58	84.23	56.13
MobileViT-XXS	68.77	88.75	71	83.72	44.52
MobileViTv2-0.5	70.15	88	70.52	79.64	35.33
SqueezeNet v1.1	58.1	89.32	69.4	80.22	10
PVTv2-B0	70.65	89.31	88.08	**84.6**	62.27
TinyViT-5M	**80.86**	**91.11**	81.6	77.75	42.33
EfficientFormerV2-S0	74.08	88.73	66.09	N/A	N/A
EfficientNet-B0	77.64	83.62	79.81	79.84	41.97
ShuffleNetV1 0.5x	56.95	83.62	77.92	72.7	28.28
ShuffleNetV2 0.5x	60.56	87.07	72.68	77.62	40.62
FastViT-T8	76.17	90.21	54.79	78.71	46.68

TABLE II: Accuracy on the test datasets (in all resolution formats) for the smallest variants of each network family. Missing values are for those networks whose original implementation didn't account for multiple input sizes. Top-1 accuracy is reported on ImageNet.

As shown in Figs. 2.a and 2.e, when matched for the same number of parameters, transformer-based models outperform CNNs on ImageNet and CIFAR-10, while achieving comparable accuracy on VWW. For the latter (Fig. 2.c) the absence of a clear winner can be attributed to the task's relative simplicity, for which most models achieve strong performance without requiring highly expressive architectures or large model capacity.

In contrast, for ImageNet and CIFAR-10, a more pronounced separation emerges. Among models with approximately one million parameters, the best-performing Attention model, EdgeNeXt-XXS, achieves an accuracy of 84.58% on CIFAR-10. In contrast, the CNN counterpart, ShuffleNetV1 $0.5\times$, reaches 77.91%, corresponding to a 6.67% accuracy gap in favor of transformers. A similar trend is observed on ImageNet, where the performance difference between transformer-based and CNN models reaches 3.81% at the same parameter budget and further increases for larger models.

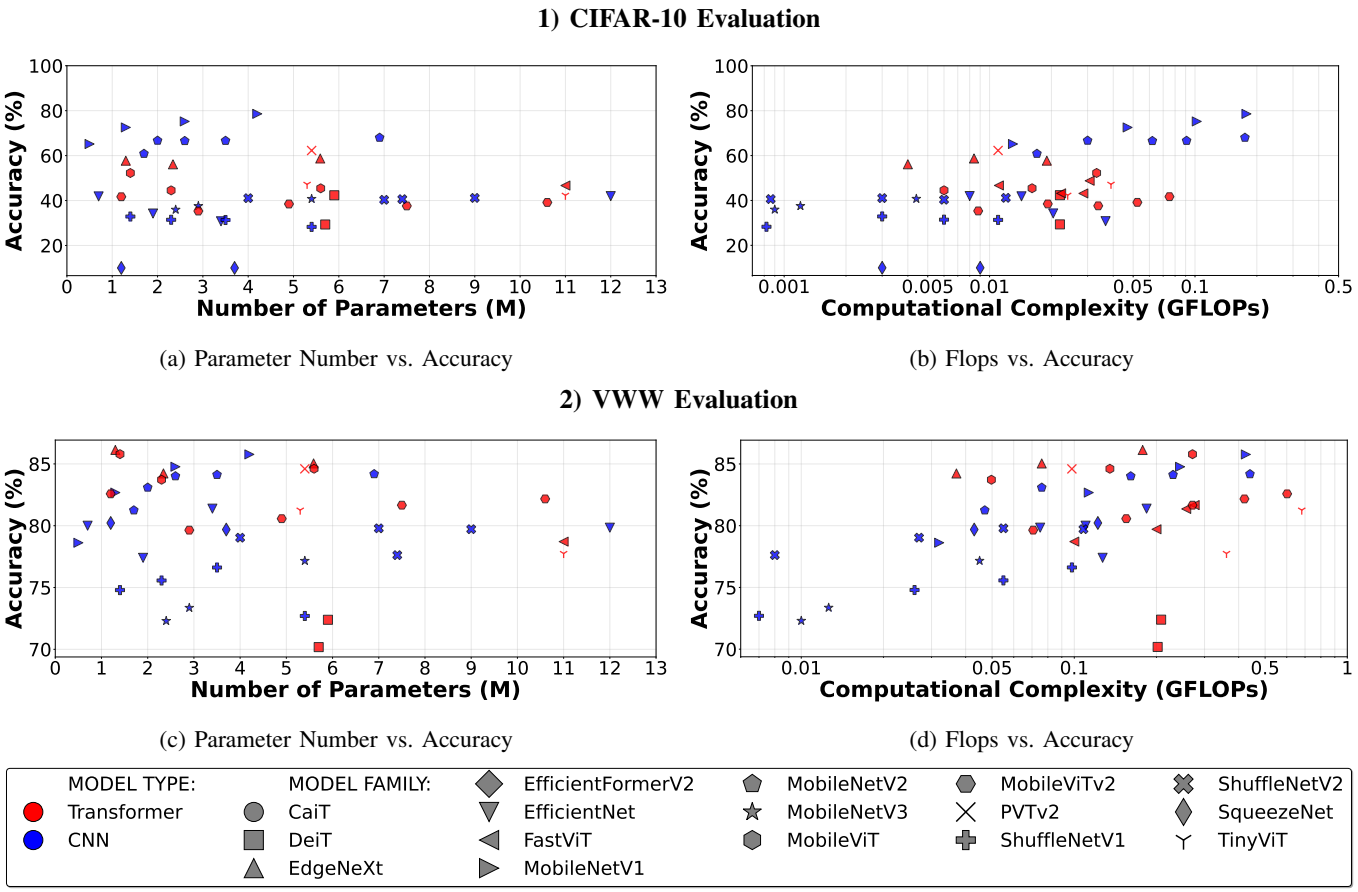
However, as shown in Fig. 2.b and Fig. 2.f, this accuracy advantage comes at the cost of higher computational complexity. Specifically, for a comparable level of accuracy, transformer-based models require up to $\sim 10\times$ as many FLOPs per input as CNNs, highlighting that, despite the higher accuracy ceiling achieved by transformers, CNNs remain significantly more computationally efficient.

B. Compressed Input Tradeoffs

We analyze robustness to changes in input resolution by training and evaluating on VWW and CIFAR-10 at input sizes smaller than those used during ImageNet pre-training (Fig. 3).

For VWW (Fig. 3.c) no clear winner emerges among the evaluated architectures. This outcome is consistent with the observations in Section V-A, as the task's relative simplicity limits the advantage of models with better adaptability to change in put size, leading to comparable performance across architectures.

In contrast, CIFAR-10 shows a more precise separation between architectures. As shown in Fig. 3.a, at 32×32 input



(a) Parameter Number vs. Accuracy

(b) Flops vs. Accuracy

(c) Parameter Number vs. Accuracy

(d) Flops vs. Accuracy

Fig. 3: Accuracy vs. number of parameters and floating point operations on the VWW and Cifar-10 datasets, with input sizes 96×96 and 32×32.

resolution and in the 1M-parameter regime, the best CNN achieves an accuracy of 72.54%, whereas the best Attention-based model reaches only 56.13%. This substantial performance gap indicates that reducing the input resolution leads to a pronounced degradation in transformer accuracy, resulting in the loss of the previously observed performance advantage. Overall, this behavior suggests that Attention-based models are less robust to changes in input resolution than CNNs, likely because they lack strong inductive biases that support effective scale adaptation.

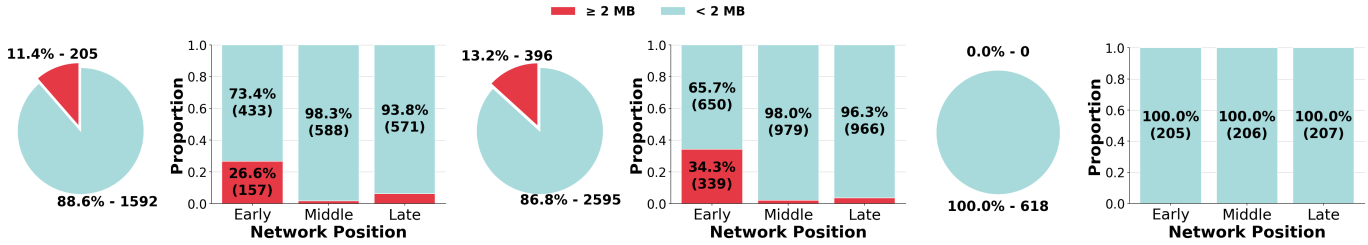
C. Working Layer Memory Distribution Analysis

Figs. 4.a, 4.b, and 4.c report, for CNN, hybrid, and transformer architectures, respectively, for an input size of 224×224, the percentage of layers whose memory footprint exceeds the 2 MiB threshold, corresponding to the available L2 memory on the target device. Each figure includes both an aggregate view and a positional breakdown along the network's topological order, with layers grouped into early, intermediate, and late stages. Considering the full network depth, 11.4% of layers in CNNs exceed the 2 MiB limit of our architecture on-chip memory, while this fraction increases to 13.2% for hybrid models; transformer-based architectures, instead, exhibit no layers above the threshold across all the analyzed networks. The concentration of critical layers is particularly pronounced in the early stages: restricting the analysis to the first 33% of the network increases the fraction of layers exceeding 2 MiB to 26.6% for CNNs and 34.4% for hybrids. Overall, these results indicate that CNN and hybrid architectures exhibit a less uniform memory footprint along depth, with pronounced peaks, especially in the early layers, whereas transformers display a more stable, predictable memory profile. This regularity makes transformer models more amenable to deployment on memory-constrained platforms, as it facilitates buffer reuse and simplifies memory management.

D. Deployment Latency Analysis

Based on the worst-case working-memory footprint, measured when processing 224×224 ImageNet inputs, and the device's 2 MiB on-chip memory budget, we restricted deployment to the models reported in Table III, ensuring that all results reflect realistic, deployable operating conditions.

Within this subset, Table III indicates that FLOPs correlate with latency only weakly: computational cost is directionally informative, but it is not a reliable predictor of cycle count on the target edge platform as effective speed depends on factors beyond arithmetic complexity (e.g., operator mix, memory access patterns, and hardware specialization). For example, CaiT



(a) CNNs memory distribution (b) Hybrid networks memory distribution (c) Transformer networks memory distribution

Fig. 4: For each subfigure: (Left) Layer-wise distribution of required working memory (sum of input activations, output activations, and parameters), grouped by whether it exceeds or fits within the OpenPULP internal memory capacity. (Right) The same metric stratified by layer position in the network's topological order: Early (first 33% of layers), Mid (33–67%), and Late (last 33%), highlighting how memory occupation evolves from front to back of the model. All results are reported for 224×224 input size, and the absolute values in the sub-figures represent the number of layers for that category.

Model	Working Mem. (MiB)	Cycles (G)	FLOPS (M)	ImageNet Acc.
CaiT-XXS-24	**1.28**	9.41	2,500	**0.78**
EdgeNeXt-XS	1.93	4.72	538	0.75
DeiT-Ti	1.29	40.43	1,300	0.74
EdgeNeXt-XXS	1.44	2.06	261	0.71
MobileNet v3 Small 0,75x	1.69	1.46	44	0.65
ShuffleNetV1 0,5x (g=3)	1.84	**1.37**	**41**	0.57
MobileNet v1 0,25x	1.53	1.55	44	0.51

TABLE III: Model memory and computational cost for 224×224 inputs.

is 4× faster than DeiT, even though DeiT reports about half the FLOPs, illustrating that nominal arithmetic complexity does not capture deployment-relevant bottlenecks. More broadly, when comparing CaiT to the simpler CNNs, one might expect a 61× latency increase if FLOPs were the dominant driver; instead, the fastest network (ShuffleNetV1 0.5×) has ≈61× fewer FLOPs than CaiT but achieves only ≈7× fewer cycles. Further demonstrating the limited usefulness of FLOPs as a stand-in for on-device latency in this regime.

Table III also highlights the working-memory advantage of Transformer architectures over both hybrid and convolutional baselines. In particular, the Transformer models require 1.13× and 1.2× less working memory than the best-performing hybrid networks and CNNs, despite being substantially larger, corresponding to 9.2× and 17.2× increases in parameter count, respectively. This has direct deployment implications: because Transformers fit more comfortably within the 2 MiB operational budget, they leave more headroom for buffering and kernel-level tuning, while achieving better accuracy under the same constraint, +21% and +7% over the smallest-footprint CNNs and hybrid models, respectively.

Finally, we notice that CNNs still exhibit a clear advantage in cycle count: even the fastest hybrid model is 1.5× slower than the fastest convolutional baseline. A likely contributing factor is that the evaluated toolchains and target platforms remain convolution-centric, meaning convolution kernels benefit from more mature operator implementations and hardware-software co-optimization, whereas transformer and hybrid networks' kernels are comparatively less optimized.

VI. Conclusion

This work presented a systematic comparison of tiny CNNs, Transformers, and Hybrid architectures for image classification on extreme-edge devices, jointly assessing accuracy and efficiency. On the accuracy side, attention-based models are consistently more effective both at acquiring in-domain knowledge and at retaining performance under semantic domain shift, achieving up to +3.81% higher in-domain accuracy and +6.67% under domain shift. However, this accuracy gain is not stable across input scales: at reduced input resolutions the advantage disappears, and CNNs surpass transformers by up to 16.4%, suggesting that transformers are comparatively more robust to semantic variation while being more sensitive to changes in input size.

To measure efficiency, we profile a subset of networks on an 8-core RISC-V microcontroller. Transformers exhibit a more predictable, better-behaved working-memory profile: their layer-wise footprint is more uniform, and no layer exceeds the 2 MiB budget, whereas 11.4% of CNN layers exceed this threshold, complicating buffer sizing and memory management. Despite this regularity, CNNs still achieve approximately a 1.5× latency advantage, plausibly due to more mature, better-optimized kernel support for convolutional operators on this platform. Overall, transformers currently lag behind CNNs in latency on extreme-edge deployments, but their higher accuracy and more bounded memory profile are strong practical benefits and indicate substantial promise as software and kernel support will reduce the latency gap.

References

[1] Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than cnns? *Advances in neural information processing systems*, 34:26831–26843, 2021.

[2] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, et al. Mlperf tiny benchmark. *arXiv preprint arXiv:2106.07597*, 2021.

[3] Nazareno Bruschi, Germain Haugou, Giuseppe Tagliavini, Francesco Conti, Luca Benini, and Davide Rossi. Gvsoc: A highly configurable, fast and accurate full-platform simulator for risc-v based iot processors. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 409–416, 2021.

[4] Alessio Burrello, Moritz Scherer, Marcello Zanghieri, Francesco Conti, and Luca Benini. A microcontroller is all you need: Enabling transformer execution on low-power iot endnodes. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2021.

[5] Santiago Canchila, Carlos Meneses-Eraso, Javier Casanoves-Boix, Pascual Cortes-Pellicer, and Fernando Castello-Sirvent. Natural language processing: An overview of models, transformers and applied practices. *Computer Science and Information Systems*, 21:1097–1145, 01 2024.

[6] Aakanksha Chowdhery, Pete Warden, Jonathon Shlens, Andrew Howard, and Rocky Rhodes. Visual wake words dataset. *arXiv preprint arXiv:1906.05721*, 2019.

[7] Francesco Conti. Technical report: Nemo dnn quantization for deployment model. *arXiv preprint arXiv:2004.05930*, 2020.

[8] Alberto Dequino, Luca Bompani, Luca Benini, and Francesco Conti. Optimizing bfloat16 deployment of tiny transformers on ultra-low power extreme edge socs. *Journal of Low Power Electronics and Applications*, 15(1):8, 2025.

[9] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[11] facebookresearch. fvcore library. https://github.com/facebookresearch/fvcore. Accessed: 2026-01-29.

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013.

[13] Vikas Hassija, Balamurugan Palanisamy, Arpita Chatterjee, Arpita Mandal, Debanshi Chakraborty, Amit Pandey, GSS Chalapathi, and Dhruv Kumar. Transformers for vision: A survey on innovative methods for computer vision. *IEEE Access*, 2025.

[14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[15] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[17] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR, 2021.

[18] Brett Koonce. Resnet 50. In *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pages 63–72. Springer, 2021.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[20] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16889–16900, 2023.

[21] Yinan Liang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Mcuformer: Deploying vision tranformers on microcontrollers with limited memory. *Advances in Neural Information Processing Systems*, 36:8501–8512, 2023.

[22] Alessandro Lotti, Dario Modenini, and Paolo Tortora. Investigating vision transformers for bridging domain gap in satellite pose estimation. In *International Conference on Applied Intelligence and Informatics*, pages 299–314. Springer, 2022.

[23] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[24] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: Efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *European Conference on Computer Vision (ECCV) Workshops*, pages 3–20. Springer, 2022.

[25] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

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

[27] Davide Nadalini, Manuele Rusci, Elia Cereda, Luca Benini, Francesco Conti, and Daniele Palossi. Multi-modal on-device learning for monocular depth estimation on ultra-low-power mcus. *IEEE Internet of Things Journal*, 2025.

[28] Davide Nadalini, Manuele Rusci, Giuseppe Tagliavini, Leonardo Ravaglia, Luca Benini, and Francesco Conti. Pulp-trainlib: Enabling on-device training for risc-v multi-core mcus through performance-driven autotuning. In Alex Orailoglu, Marc Reichenbach, and Matthias Jung, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 200–216, Cham, 2022. Springer International Publishing.

[29] Francesco Paissan, Davide Nadalini, Manuele Rusci, Alberto Ancilotto, Francesco Conti, Luca Benini, and Elisabetta Farella. Structured sparse back-propagation for lightweight on-device continual learning on microcontroller units. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2172–2181, 2024.

[30] Arpan Suravi Prasad, Moritz Scherer, Francesco Conti, Davide Rossi, Alfio Di Mauro, Manuel Eggimann, Jorge Tomás Gómez, Ziyun Li, Syed Shakib Sarwar, Zhao Wang, et al. Siracusa: A 16 nm heterogenous risc-v soc for extended reality with at-mram neural engine. *IEEE Journal of Solid-State Circuits*, 59(7):2055–2069, 2024.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[33] Moritz Scherer, Michele Magno, Jonas Erb, Philipp Mayer, Manuel Eggimann, and Luca Benini. Tinyradarnn: Combining spatial and temporal convolutional neural networks for embedded gesture recognition with short range radars. *IEEE Internet of Things Journal*, 8(13):10336–10346, 2021.

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

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

[36] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 32–42, 2021.

[37] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5785–5795, 2023.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[39] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational visual media*, 8(3):415–424, 2022.

[40] Zeyu Wang, Yutong Bai, Yuyin Zhou, and Cihang Xie. Can cnns be more robust than transformers? *arXiv preprint arXiv:2206.03452*, 2022.

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

[42] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, pages 68–85. Springer, 2022.

[43] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.