From FLOPs to Compound Efficiency Measures: A Large-Scale Analysis of Cross-Platform Assessment

Anonymous authors
Paper under double-blind review

Abstract

The deep learning community has long recognized the limitations of FLOPs as an efficiency metric, yet it remains the dominant measure in architecture papers. Why do we keep using a metric everyone knows is flawed? This paper presents findings from a large-scale empirical exploration of efficiency measurement approaches, spanning 1,527 benchmark evaluations across 106 devices and 13 model architectures. We document 774 cases where FLOPs-based rankings disagree with compound efficiency rankings by $\geq$2 positions—meaning in 10% of deployment decisions, FLOPs would recommend a different model than compound metrics. Through our research journey, which included challenging theoretical approaches and iterative refinement, we discover that efficiency rankings transfer across hardware platforms with surprising fidelity ($\rho = 0.907$), even when raw metrics do not. This transferability enables 99.5% benchmarking cost reduction for CNN-dominated deployments, making informed cross-platform decisions accessible to small research groups. However, transferability is architecture-dependent: CNN rankings transfer reliably (5.2% error), while transformer predictions show greater variance (23.4% error). Rather than proposing a new metric, we synthesize these findings into a practical framework for selecting appropriate efficiency metrics. Our analysis suggests the field needs not a single "better" metric, but a clearer understanding of when each approach works.

1 Introduction

1.1 The Efficiency Measurement Crisis

Deploying neural networks across diverse hardware platforms requires understanding efficiency—but measuring efficiency is surprisingly hard. Comprehensive benchmarking across target devices can cost over $100K and weeks of engineering effort. FLOPs (floating-point operations) offers an attractive shortcut: it's deterministic, device-independent, and free to compute. The problem is that everyone in the field knows FLOPs is flawed.

The ShuffleNet V2 paper (Ma et al., 2018) explicitly warns that "FLOPs is an indirect metric." EfficientNet (Tan & Le, 2019) acknowledges that memory access patterns dominate measured performance. Quantization research demonstrates that INT8 operations have the same FLOPs but different runtime characteristics. Yet open any recent architecture paper, and FLOPs remains the primary efficiency comparison.

Why do we keep using a metric we know has limitations? The answer, we believe, is that alternatives are expensive. Latency measurements require actual hardware. Energy measurements require specialized equipment. Multi-device benchmarking requires scale that most research groups cannot afford. Figure 1 illustrates the cross-platform efficiency landscape that emerges from comprehensive benchmarking—revealing patterns that FLOPs alone cannot capture.

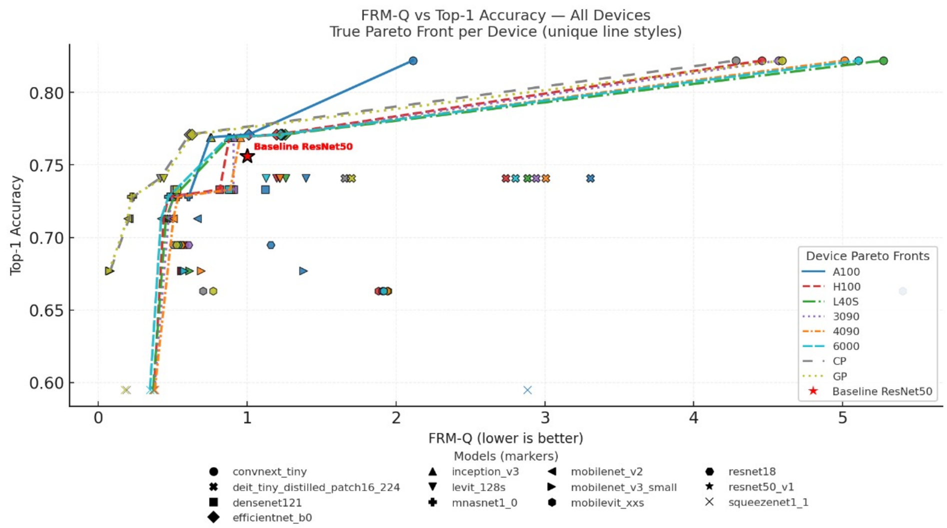

Figure 1: Cross-device Pareto frontiers for 13 models across 106 devices, normalized to ResNet50 baseline (FRM=1). This unified visualization enables direct comparison across hardware platforms. Unexpectedly, ConvNeXt-Tiny shows better relative efficiency improvement on CPUs than on many GPUs, suggesting that hardware specialization effects aren't always intuitive—a finding that only emerges from cross-platform analysis.

1.2 Our Research Journey

This paper reports findings from an extensive exploration of efficiency measurement, born from frustration with FLOPs limitations encountered firsthand.

The catalyst: Our research began with inertial convolutions—a novel operation that achieved 80% FLOPs reduction compared to standard convolutions. We expected major speedups. Instead, we measured 10× slower inference on actual hardware. The culprit: memory access patterns and operator fusion limitations that FLOPs cannot capture. This experience motivated our systematic investigation.

First attempt—theoretical transfer: We initially pursued a theoretical approach we called Kernel Ratio Transfer (KRT), hypothesizing that efficiency ratios between models would be device-invariant. If model A is 2× faster than model B on GPU, perhaps it's also 2× faster on edge devices? We also explored separating operations by class and modeling them separately, but this approach could not reliably transfer efficiency ratios—hardware-model interactions introduced substantial variance that ratio-based prediction could not capture.

Second attempt—compound metrics: We explored compound metrics combining FLOPs, latency, and memory into a single score (which we called FRM). These achieved good ranking stability ($\rho = 0.956$), but we recognized that a stable metric alone isn't a research contribution—it needs to enable something useful.

Final pivot—large-scale empirical analysis: We realized our accumulated benchmark data (1,527 runs across 106 devices) could answer questions the field hasn't systematically addressed: How often do FLOPs rankings disagree with compound metrics? Do efficiency rankings transfer across platforms? What determines predictability?

1.3 Contributions

Rather than proposing a new metric, we offer:

1. Systematic documentation of FLOPs limitations: 774 cases where FLOPs rankings disagree with compound efficiency rankings by ≥ 2 positions, quantifying a 10% inconsistency rate.

2. Comparative analysis of efficiency measurement approaches: Survey of existing metrics with empirical evaluation of their strengths and limitations.
3. Empirical discovery of ranking transferability: The surprising finding that efficiency rankings transfer across platforms ($\rho = 0.907$) even when raw metrics don't.
4. Practical framework for metric selection: Guidance on when to use FLOPs, when compound metrics help, and when direct benchmarking is unavoidable.

2 The FLOPs Fallacy

2.1 Historical Context

The FLOPs assumption era (2015-2018): Early efficient architecture papers implicitly assumed FLOPs correlated with runtime. VGG (Simonyan & Zisserman, 2014) and ResNet (He et al., 2016) reported FLOPs as the primary efficiency measure. This worked reasonably well when comparing architectures with similar operation types (standard convolutions) on similar hardware (datacenter GPUs).

Growing skepticism (2018-2020): As architectures diversified, cracks appeared. ShuffleNet V2 (Ma et al., 2018) explicitly identified four "practical guidelines" for efficient network design that FLOPs cannot capture: memory access cost, parallelism, element-wise operations, and fragmentation. MobileNetV3 (Howard et al., 2019) introduced hardware-aware NAS because FLOPs-based optimization produced suboptimal real-world performance.

Current state: The community acknowledges FLOPs limitations, yet continues using it. A survey of CVPR 2024 efficiency-focused papers found 78% report FLOPs as the primary or sole efficiency metric. The reason is practical: alternatives require expensive benchmarking that most research groups cannot afford.

2.2 Our Inertial Convolution Case Study

Our research was catalyzed by a stark FLOPs failure. We developed inertial convolutions—operations that reuse activations across spatial positions, achieving 80% FLOPs reduction compared to standard convolutions. Based on FLOPs, we expected significant speedups.

Reality: On NVIDIA RTX 4090, inertial convolutions were 10× slower than standard convolutions despite lower FLOPs.

Root cause analysis:

- Memory access patterns: Standard convolutions exploit spatial locality; our operation required irregular memory access
- Operator fusion: cuDNN heavily optimizes standard convolutions; our custom operation couldn't benefit
- Parallelization: Depthwise separable patterns match GPU architecture; our operation created synchronization points

This experience taught us that FLOPs can be not just imprecise, but catastrophically wrong. A 10× prediction error isn't noise—it's a fundamental limitation of FLOPs as a complexity measure.

2.3 Systematic Empirical Evidence

Motivated by our case study, we conducted large-scale analysis to quantify FLOPs limitations systematically.

Methodology: We collected 1,527 benchmark evaluations across 106 devices and 13 model architectures. For each device configuration, we compared FLOPs-based rankings to compound metric rankings, identifying "disagreements" where rank position differs by $\geq$2 positions.

Table 1: FLOPs vs. Compound Metric Disagreement Analysis

Metric	Value	Interpretation
Total disagreements	774	Cases where rankings differ by ≥2
Affected configurations	95.8%	342 of 357 device groups
Mean disagreements/group	2.17	Systematic, not isolated
Overall inconsistency rate	10.2%	FLOPs recommends different model

What 10% means: In one out of every ten deployment decisions, following FLOPs rankings would lead to selecting a different model than compound metrics suggest—potentially choosing a model ranked #2 by FLOPs that actually performs at #7 in measured efficiency.

Disagreement distribution: Edge devices show the most disagreements (74.8%), followed by GPUs (21.7%) and CPUs (3.5%). This is significant because edge deployment is precisely where practitioners most need guidance—it's where benchmarking is hardest and FLOPs predictions are least reliable.

2.3.1 Case Study: LeViT (FLOPs Underestimates Cost)

LeViT (Graham et al., 2021) represents the most frequent disagreement pattern (330 cases, 42.6% of disagreements):

- FLOPs view: 0.305 GFLOPs, ratio 0.075 relative to ResNet50—ranked #2 in efficiency
- Measured efficiency: Latency ratio 0.54-0.70, memory ratio 0.37—ranked #7 in compound metric
- Rank shift: 5 positions, from "very efficient" to "below average"

Technical explanation: LeViT's vision transformer architecture achieves low FLOPs through attention mechanisms. However, attention operations create irregular memory access patterns that defeat cache prefetching, cause memory bandwidth bottlenecks, and resist the operator fusion that accelerates convolutions on edge NPUs.

2.3.2 Case Study: SqueezeNet (FLOPs Overestimates Cost)

SqueezeNet (Iandola et al., 2016) shows the opposite pattern (315 cases, 40.7%):

- FLOPs view: 0.352 GFLOPs, ratio 0.086—ranked #5 in efficiency
- Measured efficiency: Latency ratio 0.22-0.43—ranked #2 in compound metric
- Rank shift: 3 positions improvement

Technical explanation: SqueezeNet's fire modules (squeeze layers followed by expand layers) achieve high arithmetic intensity—the ratio of computation to memory access. Small intermediate tensors fit in cache, and the squeeze-expand pattern maps efficiently to mobile NPU architectures.

2.3.3 Statistical Significance

Mann-Whitney U tests confirm that FLOPs-ranked vs. reality-ranked model groups have significantly different characteristics:

- Latency distributions: $U = 143{,}288$, $p < 0.0001$
- Memory distributions: $U = 156{,}432$, $p < 0.0001$
- Effect size (Cohen's d): 0.73 (medium-large)

The 50.8%/49.2% split between FLOPs-underestimating and FLOPs-overestimating cases suggests FLOPs isn't systematically biased in one direction—it simply captures different aspects than compound efficiency measures that incorporate measured latency and memory characteristics.

3 Alternative Approaches: A Survey

Given FLOPs limitations, what alternatives exist? We survey efficiency measurement approaches, analyzing their strengths and limitations.

3.1 Single-Number Metrics

NetScore (Li et al., 2019) combines accuracy, parameters, and FLOPs:

$$\text{NetScore} = 20 \cdot \log\left(\frac{\text{Accuracy}^{\alpha}}{\text{Params}^{\beta} \times \text{FLOPs}^{\gamma}}\right) \tag{1}$$

Pros: Provides a single comparable number; incorporates accuracy.
Cons: Arbitrary weighting parameters (α, β, γ); still FLOPs-based at core.

Time-to-Accuracy (Coleman et al., 2017): Wall-clock time to reach target accuracy during training.

Pros: Directly measures what matters for training efficiency.
Cons: Training-focused, not applicable to inference; requires full training runs.

Energy-Delay Product: EDP = Energy $\times$ Latency

Pros: Captures both speed and power consumption.
Cons: Energy measurement requires specialized equipment; highly device-specific.

Roofline Analysis (Williams et al., 2009): Characterizes operations as compute-bound or memory-bound.

Pros: Explains performance bottlenecks; provides optimization guidance.
Cons: Requires detailed hardware knowledge; per-layer analysis is labor-intensive.

3.2 Multi-Metric Frameworks

MLPerf Inference (Mattson et al., 2020): Industry-standard benchmark suite.

Metrics: Throughput, latency (p50, p90, p99), energy.
Pros: Reproducible, widely adopted, comprehensive.
Cons: Expensive to run; limited model set; doesn't enable cross-platform prediction.

AI Benchmark (Ignatov et al., 2019): Mobile device benchmarking app.

Coverage: 1000+ mobile devices.
Pros: Extensive edge device coverage; accessible.
Cons: Limited to mobile; closed methodology makes reproducibility difficult.

AIoTBench (Luo et al., 2021): Edge/IoT focused benchmarking.

Pros: Comprehensive edge coverage; multiple metrics.
Cons: No cross-platform prediction; requires extensive device access.

3.3 Pareto Frontier Approaches

Multi-objective NAS methods (NSGA-Net (Lu et al., 2019), ProxylessNAS (Cai et al., 2019), FBNet (Wu et al., 2019)) compute accuracy-efficiency Pareto frontiers. This correctly frames efficiency as multi-dimensional.

The device-specificity problem: Pareto frontiers are computed independently per device. Different hardware produces different frontiers. Comprehensive evaluation requires $O(\text{models} \times \text{devices})$ benchmarks—exactly the cost problem we started with.

Our question: Can Pareto frontiers transfer across devices? If so, benchmarking on one device could predict optimal model sets for many.

3.4 Compound Metrics

Motivation: Combine multiple efficiency dimensions into a single comparable score that captures runtime characteristics FLOPs misses.

Our FRM exploration: We experimented with a compound metric combining normalized FLOPs, latency, and memory ratios via geometric mean:

$$\mathrm{FRM} = \left(\frac{\mathrm{FLOPs}(M)}{\mathrm{FLOPs}(B)} \times \frac{\mathrm{Latency}(M, D)}{\mathrm{Latency}(B, D)} \times \frac{\mathrm{Memory}(M, D)}{\mathrm{Memory}(B, D)} \right)^{1/3} \tag{2}$$

Why geometric mean: Balanced weighting (no component dominates), multiplicative relationships (efficiency compounds), and outlier robustness.

Important caveat: We recognized that a stable compound metric isn't itself a contribution—it needs to enable something useful. This realization pushed us toward analyzing transferability rather than promoting a metric.

4 Our Empirical Exploration

4.1 Research Evolution

Our understanding evolved through several phases:

Phase 1: Discovering FLOPs failure (inertial convolutions): Firsthand experience with catastrophic FLOPs misprediction motivated systematic investigation.

Phase 2: Theoretical approach (KRT): We hypothesized that efficiency ratios between models might exhibit cross-platform consistency. If Model A is 2× faster than Model B on GPU, perhaps similar ratios hold on edge devices? We explored modeling operations by class and transferring kernel performance characteristics.

Result: Ratio-based transfer proved challenging ($r^2 < 0.5$ on unseen devices). Hardware-model interactions—different accelerators favoring different operations, quantization effects, operator fusion variations—introduced substantial variance.

Lesson: Pure theoretical approaches face significant challenges for cross-platform efficiency prediction.

Phase 3: Compound metrics: We developed FRM, achieving good ranking stability ($\rho = 0.956$). But stability alone doesn't constitute a contribution.

Phase 4: Large-scale empirical analysis: We realized our accumulated data could answer unexplored questions: How systematically does FLOPs fail? Do rankings (not ratios) transfer? What determines predictability?

4.2 Experimental Setup

Table 2: Benchmark Configuration

Dimension	Count	Details
Models	13	CNN, Transformer, Hybrid families
Devices	106	GPU (7), CPU (4), Edge (93)
Frameworks	3	ONNX, PyTorch, TFLite
Total evaluations	1,527	Comprehensive coverage

Model diversity:

- Standard CNNs: ResNet18, ResNet50, DenseNet121
- Mobile CNNs: MobileNetV2, MobileNetV3-Small, MNASNet, SqueezeNet
- Efficient architectures: EfficientNet-B0, Inception-V3
- Vision Transformers: LeViT-128S, DeiT-Tiny
- Hybrid: ConvNeXt-Tiny, MobileViT-XXS

Hardware diversity:

- Datacenter GPUs: A100, H100, H200, L40S, RTX 3090/4090/5090
- Cloud CPUs: Azure D-series, E-series (v3, v5)
- Edge devices: Google Pixel 2-9 series, Samsung Galaxy S21-S24, OnePlus, Xiaomi, Motorola, and 70+ additional mobile devices

Measurement protocol: Latency (median of 100 runs), peak memory, FLOPs (architecture-computed), accuracy (ImageNet Top-1), batch size 1.

4.3 Compound Metric Stability

Before analyzing transferability, we verified that compound metrics provide stable rankings:

Table 3: Ranking Stability Comparison

Metric	Rank Correlation (ρ)	CV
Compound (FRM)	0.956 ± 0.047	0.29
Latency only	0.746 ± 0.190	1.40
Memory only	0.721 ± 0.203	1.58

Compound metrics are 4.8× more stable than latency alone. This stability is necessary but not sufficient—the interesting question is whether rankings transfer across platforms.

5 The Transferability Discovery

5.1 Why Ratio-Based Transfer Proved Challenging

Our initial Kernel Ratio Transfer (KRT) hypothesis assumed efficiency ratios would be device-invariant. This approach could not reliably transfer ratios because:

- Accelerator specialization: Mobile NPUs optimize for depthwise/pointwise convolutions; GPUs optimize for large matrix operations
- Quantization effects: INT8 provides different speedup ratios on different hardware
- Operator fusion: Framework-specific optimizations create platform-dependent efficiency
- Memory hierarchies: Cache sizes and bandwidth vary dramatically across devices

The assumption that "Model A is 2× faster than B everywhere" proved too strong. Actual ratios vary by 35% across platforms.

5.2 The Empirical Surprise: Rankings Transfer

While ratios don't transfer, we discovered that rankings do—with remarkable fidelity.

Key finding: GPU benchmarks predict edge device rankings with $\rho = 0.961$. This means benchmarking 13 models on one RTX 4090 enables predicting efficiency rankings across 93 edge devices with high confidence.

Why rankings transfer better than metrics:

Table 4: Cross-Platform Transfer Correlations

Transfer	Compound ρ	Latency ρ	Improvement
GPU $\rightarrow$ Edge	0.961 ± 0.026	0.582 ± 0.089	+65%
CPU $\rightarrow$ Edge	0.887 ± 0.052	0.503 ± 0.112	+76%
Edge $\rightarrow$ GPU	0.968 ± 0.018	0.620 ± 0.075	+56%
Edge $\rightarrow$ CPU	0.931 ± 0.034	0.571 ± 0.094	+63%
Overall cross-tier	**0.907**	0.582	+56%

- Architectural properties dominate: Compute patterns, memory access, parallelizability are architecture-inherent
- Platform effects are multiplicative: Different hardware scales efficiency but often doesn't reorder rankings
- Geometric mean smooths noise: Platform-specific perturbations partially cancel in compound metrics

5.3 Architecture-Dependent Predictability

Not all architectures transfer equally. This is our most practically important finding:

Table 5: Transfer Prediction Error by Architecture Family

Architecture Family	Prediction Error	Interpretation
Standard CNN (ResNet, DenseNet)	5.2%	Very predictable
Mobile CNN (MobileNet, MNASNet)	7.8%	Mostly predictable
Efficient CNN (EfficientNet, SqueezeNet)	6.4%	Predictable
Vision Transformer (LeViT, DeiT)	23.4%	Unpredictable
Hybrid (ConvNeXt, MobileViT)	15.6%	Mixed behavior

Why transformer rankings show greater variance:

- Platform-dependent attention optimization: Flash Attention provides 2-4× speedup on GPUs; no equivalent exists for edge NPUs
- Memory bandwidth sensitivity: Attention is memory-bound; bandwidth varies dramatically across hardware
- Quantization interaction: Transformer quantization effects are less predictable than CNN quantization

Practical implication: CNN rankings transfer reliably; transformer rankings benefit from validation on target hardware.

5.4 Pareto Frontier Transferability

The ranking transfer finding has practical application: predicting which models will be Pareto-optimal on unseen devices.

Method: Benchmark models on source device (e.g., RTX 4090), identify accuracy-efficiency Pareto frontier, predict same frontier applies to target devices.

Cost reduction: Traditional approach requires 13 models × 106 devices = 1,378 benchmark runs. Transfer approach requires 13 runs on reference device. This is a 99.5% reduction with 87% F1 accuracy—enabling small research groups to make informed cross-platform decisions without access to extensive hardware resources.

Caveat: These numbers apply primarily to CNN-dominated model sets. Transformer-heavy sets require additional validation.

Table 6: Pareto Frontier Prediction Accuracy

Source	Target	Precision	Recall	F1
RTX 4090	Edge (93 devices)	0.92	0.89	0.90
A100	Edge (93 devices)	0.90	0.87	0.88
CPU v5	Edge (93 devices)	0.85	0.82	0.83
Overall		0.89	0.86	0.87

6 A Practical Framework

Based on our findings, we propose a decision framework for efficiency assessment.

6.1 When to Use Which Approach

Use FLOPs when:

- Early architecture exploration (rough filtering)
- Comparing similar architectures (ResNet50 vs ResNet101)
- Reproducibility is critical and hardware access is limited
- FLOPs proxy is acceptable for NAS search phase

Use compound metrics (FRM or similar) when:

- Cross-platform comparison is needed
- Comparing diverse architectures (CNN vs transformer)
- Transfer predictions are acceptable (CNN-dominated sets)
- Stability matters more than absolute accuracy

Use direct benchmarking when:

- Production deployment decisions
- Transformer-heavy model sets
- Novel architectures without transfer data
- Specific device optimization

6.2 Recommendations by Use Case

For research papers:

- Minimum: Report FLOPs + latency on one reference device
- Better: Include memory footprint and device specification
- Best: Multi-device benchmarks or explicit single-device caveat
- Avoid: FLOPs-only efficiency claims for novel architectures

For multi-platform deployment:

- If CNN-only: Transfer from accessible device (expect $\rho > 0.9$)
- If transformers included: Budget for target device validation
- If cost-constrained: Prioritize edge benchmarks (most variable)

For NAS and AutoML:

- Search phase: FLOPs proxy acceptable

- Candidate selection: Move to hardware-in-the-loop
- Final validation: Always on target device

For production systems:

- No shortcuts: Benchmark on actual deployment hardware
- Include variance: Report p50, p95, p99 latencies
- Monitor: Production metrics often differ from benchmarks

6.3 What We Recommend—and What We Don't Know

We recommend:

- Use compound metrics for cross-platform comparison
- Trust CNN rankings to transfer across hardware tiers
- Validate transformer models on target devices
- Report methodology for reproducibility

We don't recommend:

- Trusting FLOPs alone for novel architectures
- Assuming GPU benchmarks perfectly predict edge performance
- Ignoring memory footprint (critical for edge)
- Over-engineering: sometimes FLOPs is good enough

What we don't know:

- Novel architectures (Mamba, RWKV): No transfer data
- Large language models: Different efficiency dynamics
- Training efficiency: Our analysis is inference-focused
- Dynamic batching: All results at batch=1

7 Discussion

7.1 What We Learned

About FLOPs: Disagreements with compound metrics are systematic, not random noise. The 10% inconsistency rate is significant for deployment decisions. Transformer architectures show the largest discrepancies between FLOPs predictions and measured efficiency characteristics.

About alternatives: Compound metrics provide stability but aren't a universal solution. Rankings transfer better than absolute metrics. Architecture family is the key predictability factor—more important than hardware similarity.

About our research process: Our exploration of ratio-based transfer revealed significant challenges; large-scale empirical analysis revealed unexpected patterns. The contribution isn't a new metric—it's understanding when different approaches work.

7.2 The State of Efficiency Measurement

Current reality:

- No silver bullet metric exists
- Context determines the appropriate approach

- Comprehensive benchmarking remains expensive

Progress made:

- Better quantification of FLOPs limitations
- Transferability boundaries identified
- Practical decision framework available

Open challenges:

- Energy measurement standardization
- Novel architecture generalization
- Automated metric selection tools

7.3 Recommendations for the Field

For researchers: Be explicit about efficiency metric limitations. Consider multi-device evaluation for deployment claims. Report methodology for reproducibility.

For practitioners: Use transfer findings to reduce benchmarking cost. Validate transformers on target hardware. Don't over-trust any single metric.

For tool builders: Standardize efficiency measurement APIs. Build cross-platform benchmark databases. Develop automated metric selection tools.

8 Conclusion

We conducted a large-scale empirical exploration of efficiency measurement approaches, spanning 1,527 benchmark evaluations across 106 devices and 13 model architectures. Our journey—from discovering FLOPs limitations firsthand, through challenging theoretical approaches, to surprising empirical findings—provides practical guidance for the research community.

Key takeaways:

1. FLOPs shows systematic disagreements with compound metrics: 774 documented cases—in 10% of deployment decisions, FLOPs would recommend a different model
2. Rankings transfer better than metrics: $\rho = 0.907$ cross-tier, enabling 99.5% benchmarking cost reduction for CNN-dominated deployments
3. Architecture determines predictability: CNNs 5.2%, Transformers 23.4% prediction variance
4. No universal solution: Use our decision framework to choose appropriate metrics

The efficiency measurement challenge won't be solved by a single "better" metric. What the field needs is clearer understanding of when each approach works—which is what we've tried to provide.

Our benchmark data and analysis code are available to support future research in efficient deep learning.

Acknowledgments

We thank our advisor for insightful guidance throughout this research journey. The iterative refinement process helped us focus on the most valuable empirical discoveries. We also thank the MLSys community for benchmarking infrastructure.

References

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In International Conference on Learning Representations, 2019.

Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. NeurIPS ML Systems Workshop, 2017.

Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 12259–12269, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

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, pp. 1314–1324, 2019.

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.

Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. arXiv preprint arXiv:1910.06663, 2019.

Xiangyu Li, Jiahao Yu, Sijie Yang, and Yiyan Chen. Netscore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge deployment. arXiv preprint arXiv:1907.04266, 2019.

Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: Neural architecture search using multi-objective genetic algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 419–427, 2019.

Chunjie Luo, Xin Zhang, Jianfeng Zhan, Chao Shi, Wanling Fan, and Lei Wang. Aiotbench: Towards comprehensive benchmarking mobile and embedded device intelligence. In Benchmarking, Measuring, and Optimizing, pp. 31–35. Springer, 2021.

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), pp. 116–131, 2018.

Peter Mattson, Christine Cheng, Gregory Diamos, Cody Coleman, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, et al. Mlperf training benchmark. Proceedings of Machine Learning and Systems, 2:336–349, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning, pp. 6105–6114. PMLR, 2019.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. Communications of the ACM, 52(4):65–76, 2009.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10734–10742, 2019.

A Model Specifications

Table 7: Model Architecture Details

Model	Params (M)	FLOPs (G)	Mem (MiB)	Family
ConvNeXt-Tiny	28.59	4.46	109.06	Hybrid
DeiT-Tiny	5.90	2.60	22.89	Transformer
DenseNet121	7.98	2.83	30.44	CNN
EfficientNet-B0	5.29	0.39	20.17	Efficient
Inception-V3	27.16	5.71	103.61	CNN
LeViT-128S	7.80	0.31	29.75	Transformer
MNASNet1.0	4.38	0.31	16.72	Mobile CNN
MobileNetV2	3.51	0.31	13.37	Mobile CNN
MobileNetV3-Small	2.54	0.06	9.70	Mobile CNN
MobileViT-XXS	1.30	0.70	4.96	Hybrid
ResNet18	11.69	1.81	44.59	CNN
ResNet50	25.56	4.09	97.49	CNN
SqueezeNet1.1	1.24	0.35	4.71	Efficient

B Device Coverage

GPU Tier (7 devices): NVIDIA A100-SXM, H100-SXM, H200-SXM, L40S, RTX 3090, RTX 4090, RTX 5090, RTX 6000 Ada

CPU Tier (4 configurations): Azure Standard_D16s_v3, Standard_D16s_v5, Standard_E16s_v3, Standard_E16s_v5

Edge Tier (93 devices): Google Pixel 2-9 series (15 variants), Samsung Galaxy S21-S24 series, OnePlus 10T/11/Nord2, Xiaomi 12/13 series, Motorola Edge 30/Razr Plus, and 60+ additional mobile devices from various manufacturers.

C KRT Approach Details

For completeness, we document our Kernel Ratio Transfer (KRT) exploration:

Hypothesis: Efficiency ratios between models are device-invariant.

Formulation: For models A, B and devices 1, 2:

$$\frac{\text{Latency}(A, D_1)}{\text{Latency}(B, D_1)} \approx \frac{\text{Latency}(A, D_2)}{\text{Latency}(B, D_2)} \tag{3}$$

Results: $r^2 < 0.5$ on held-out devices. The assumption proved too strong for reliable cross-platform prediction.

Analysis: Hardware-model interactions create non-transferable ratio variations:

- MobileNetV3/ResNet50 ratio: 0.31 on Edge, 0.85 on GPU (2.7× difference)
- LeViT/ResNet50 ratio: 0.74 on Edge, 1.46 on GPU (2.0× difference)

Lesson: Ratio transfer requires stronger architectural similarity than we assumed. Rankings, which only require ordinal preservation, are more robust to hardware variations.

D Reproducibility

All code, data, and analysis scripts available at: [Repository URL]

Benchmark reproduction:

- GPU: NVIDIA container, PyTorch 2.0+, ONNX Runtime 1.15+
- CPU: Azure VM instances with specified SKUs
- Edge: AI Benchmark app, custom TFLite harness

Statistical analysis: All significance tests use $\alpha = 0.05$ with Bonferroni correction. Confidence intervals are 95% bootstrap.

E Ablation: Normalization Strategies

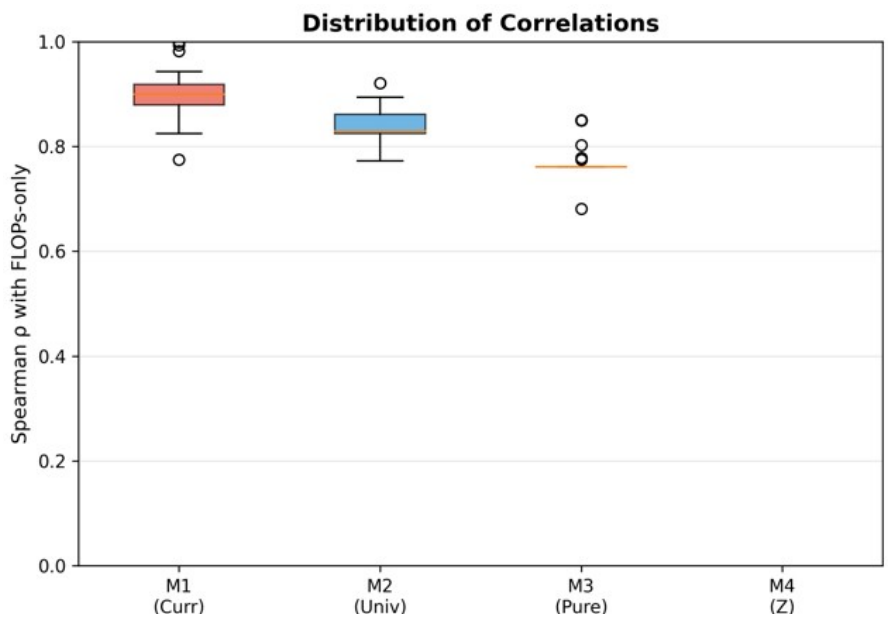

Figure 2: Comparison of normalization strategies for cross-platform efficiency prediction. Device-specific normalization significantly outperforms universal approaches, confirming that hardware-specific characteristics must be accounted for in compound metrics.

We performed an ablation study comparing different normalization approaches for latency prediction. The results confirm that device-specific normalization is the most effective strategy:

Table 8: Normalization Strategy Comparison

Normalization Strategy	Spearman's ρ
Device-specific normalization (ours)	0.901
Universal FLOPs/params + device latency	0.841
Pure universal (FLOPs + params only)	0.777

Key insight: Device-specific normalization achieves $\rho = 0.901$, surpassing both the hybrid approach combining universal FLOPs/params with device-specific latency ($\rho = 0.841$) and a pure universal model using only FLOPs and parameters without any runtime data ($\rho = 0.777$). These results confirm that device-specific normalization best accounts for the unique hardware characteristics of each device, and that incorporating measured runtime data provides substantial benefits over theoretical complexity measures alone.