Methods

- TinyML - Provides energy efficiency, responsiveness, and privacy.

  Limited by less capable ML models on resource-constrained devices, restricting use to simple tasks

- DNN - Some layers of the DNN are executed on the ED and some on the ES.

- HI - Augments on-device inference by offloading complex samples to edge servers/clouds.

  Existing studies focus on accuracy improvements but overlook latency and energy costs, and ignore device heterogeneity.

Problem Statement

Challenges:

- On-device inference faces limitations in accuracy, latency, and energy consumption as tasks grow more complex.
- Devices vary in hardware, network connectivity, and ML model capabilities.

Objectives

- Systematically compare the performance of Hierarchical Inference (HI) against on-device inference.
- Metrics: accuracy, latency, and energy consumption across different devices (Arduino Nano, ESP32, Coral Micro, Raspberry Pi 4B, Jetson Orin Nano) and datasets (MNIST, CIFAR-10, ImageNet-1K).

Existing Methods

Inference Load Balancing:

- Balances the inference task between device and server based on job execution times, communication times, energy consumption, and accuracy.
- Issue: May lead to incorrect inferences on-device in adversarial scenarios.

DNN Partitioning:

- Distributes layers of a large DL model between device and server.
- Limitations: Requires mobile GPUs; infeasible for resource-constrained devices like IoT.

On-Device Inference:

- Focused on designing compact DL models for edge devices.
- Example Models: MobileNet, Efficient Net, ResNet-8.
- Challenges: Time per inference and energy consumption not critically addressed.

Hierarchical Inference (HI):

- Combines local inference with offloading for complex samples.
- Decision based on soft-max value (confidence) from the local DL model.

HI

- **Dataset & Models:**
  - **Dataset:** CIFAR-10 (50,000 training, 10,000 test images)
  - **S-ML:** 5-layer CNN, Accuracy: 62.58%
  - **L-ML:** EfficientNet, Accuracy: 95%
- **HI Approach:**
  - **Threshold ($\theta* = 0.607$):** Offload samples if S-ML's confidence p<θp < θp<θ.

Approach	No Offload	Full Offload	Hierarchical Inference
Offloaded Images (%)	0(0%)	10000(100%)	3550(35.5%)
Misclassified Images (%)	3,742(37.42%)	500(5%)	1,577 **ON ED** + 71 **ON ES** (16.48%)
Accuracy (%)	62.58%	95%	83.52%
Cost (β)	3742	10000 β + 500	3550 β + 1648

Performance

- **Full-Offload:**
  - **Accuracy:** 95%
  - **Cost:** 10,000×β+50010,000 \times β + 50010,000×β+500
- **Local Inference:**
  - **Accuracy:** 62.58%
  - **Cost:** 3742
- **Hierarchical Inference (HI):**
  - **Accuracy:** 83.52%
  - **Cost:** 3550×β+16483550 \times β + 16483550×β+1648
  - **Cost Reduction:** 14% to 49% compared to full offload

Dog Breed Classification

Classify dog breeds in CIFAR-10 images using HI.

- **S-ML (Self-Learning Model):**
  - **Task:** Binary classification (dog vs. non-dog)
  - **Model:** 5-layer CNN, 0.23 MB, Accuracy: 63.86%
  - **Threshold:** Offload if p≥0.5p \geq 0.5p≥0.5
- **L-ML (Large Model):**
  - **Task:** Breed classification for dog images
  - **Accuracy:** 100% (idealized for comparison)

Approach	Full offload	HI
Number of Offloaded Images	10, 000	4, 433
Accuracy (%)	100%	91.2%
Cost	$1000\,\beta + 9000$	$912\,\beta + 3521$
Cost Reduction (%)	0%	$(\frac{88\beta+5479}{1000\beta+9000} \times 100)\%$

HI Approach:

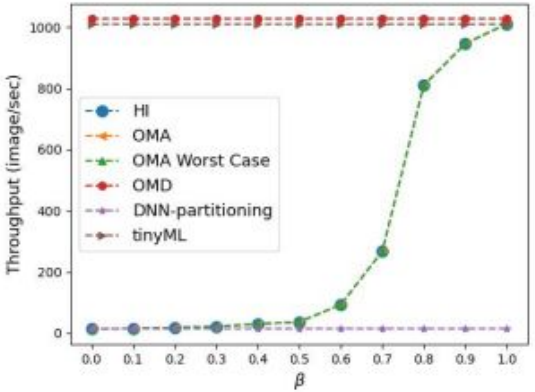
- **Dog Images:** Offloaded to ES for breed classification
- **Non-Dog Images:** Processed locally, discarded if not a dog

Results

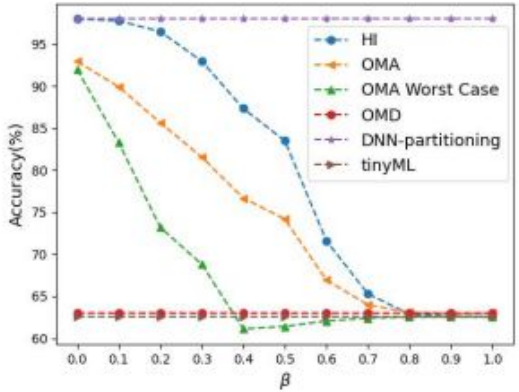
Accuracy & Costs:

- **Full Offload:**
  - **Accuracy:** Maximal
  - **Cost:** 9,000×β9,000 \times β9,000×β (includes irrelevant images)
- **Hierarchical Inference (HI):**
  - **Accuracy:** 91.2%
  - **Cost:** 88×β+5,47988 \times β + 5,47988×β+5,479
  - **Cost Reduction:** 50% - 60% compared to full offload

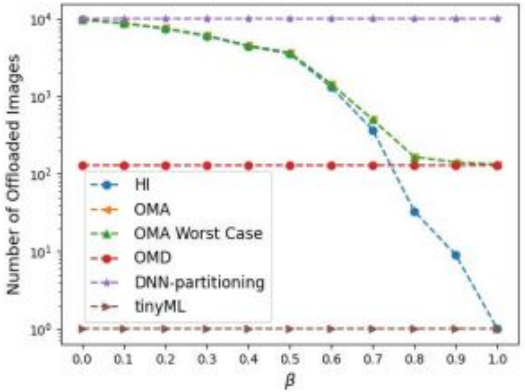
Results



(a) Throughput of CIFAR-10 image classification for different approaches.



(b) Accuracy of CIFAR-10 image classification for different approaches.



(c) Number of images offloaded for CIFAR-10 image classification for different approaches.

Methodology: Devices and Datasets

Devices:

- **Arduino Nano, ESP32:** Least powerful, support TensorFlow Lite Micro.
- **Coral Micro:** Higher processing power, supports Edge TPU but limited by supported instructions.
- **Raspberry Pi 4B:** Broad system-on-chip ecosystem, uses TFLite Python API.
- **Jetson Orin Nano:** Integrated GPU, TensorRT SDK, supports dynamic clock rates.

Datasets:

- **MNIST:** Simple task.
- **CIFAR-10:** Moderately complex.
- **ImageNet-1K:** Complex task.

Consider an image classification application1 with a QoS requirement of at least 90% accuracy and maximum 250 ms latency

Methodology: Models and Training

Base Models:

- **CIFAR-10:** ResNet-8, ResNet-56, AlexNet.
- **ImageNet-1K:** ResNet-18, ResNet-50, AlexNet, RegNetY32GF.

Early Exit (EE) Models:

- Implemented using methodology from BranchyNet.
- Thresholds set based on model confidence (highest softmax value).

Performance Metrics:

- **Energy per Inference (EPI):** Average energy consumed during an inference task.
- **Latency per Inference (LPI):** Average time taken for inference.
- **Accuracy:** Percentage of correct inferences (top-1 accuracy).

On Device Method

Model Deployment:

- **Models: Pre-trained or obtained from libraries.**
- **Quantization: INT8 used for lower latency and energy, with minor accuracy trade-offs.**

Measurement Process:

- **Latency & Energy: Measured during inference API calls, averaged over nnn images (dataset-dependent).**
- **Power Consumption: Captured using Voltech PM1000+ Power Analyzer with negligible standard deviation (max 6.1% for AlexNet on Coral Micro).**

On-Device Inference Results

Findings:

- Arduino and ESP32 can handle simple tasks (MNIST) with Logistic Regression (LR) but struggle with more complex tasks (CIFAR-10 and ImageNet-1K).
- Even Coral Micro, with its TPU, fails to meet QoS requirements (e.g., 90% accuracy, 250ms latency) for complex tasks like ImageNet-1K.

Graph: Inference accuracy vs latency for different models across devices.

Data Example: ResNet-8 on Coral Micro has 86.98% accuracy for CIFAR-10, but with higher latency and energy consumption.

Hierarchical Inference (HI)

Overview:

- HI system first performs local DL inference. If local inference is likely incorrect (based on a binary logistic regression model), the sample is offloaded to an edge server for further inference.

Equation for HI System Accuracy:

$$Acc_{HI} = \frac{\sum_{i=1}^{N} \mathbb{1}(LR_i = 1)\,\mathbb{1}(dev\_inf_i = gt_i) + \mathbb{1}(LR_i = 0)\,\mathbb{1}(ser\_inf_i = gt_i)}{N}$$

$$Time_{HI} = t_{dev\_inf} + t_{LR} + \eta_{off}\,[t_{off}]$$

$$Energy_{HI} = e_{dev\_inf} + e_{LR} + \eta_{off}\,[e_{edge\_off}]$$

EE-Hi - the early exit DL model depend on the chosen threshold θ at the early exit branch.

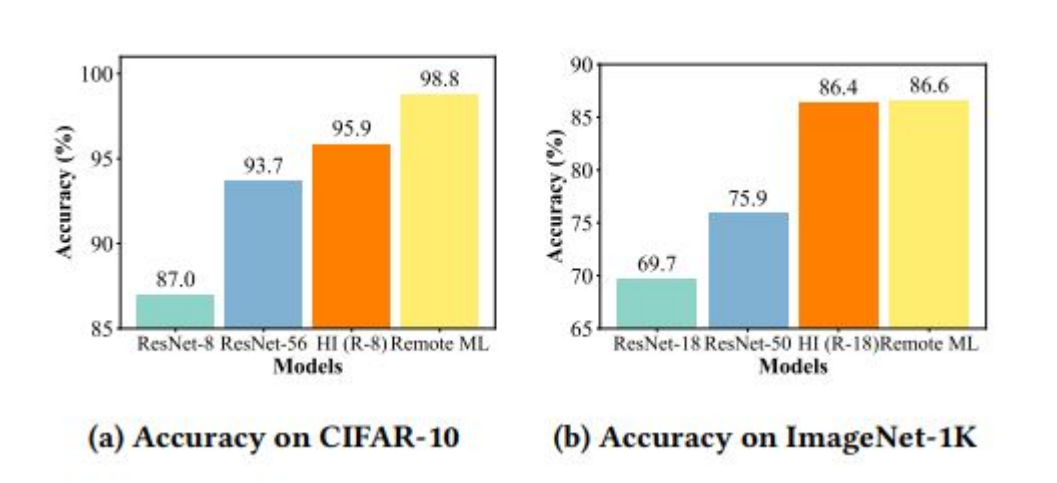
Key Results:

- HI achieves higher accuracy with minimal increases in latency and energy compared to smaller on-device models.
- Example: HI with ResNet-8 on Raspberry Pi achieves 95.9% accuracy with 73% lower latency and 77% lower energy consumption compared to on-device ResNet-56.

Measurement results and comparison

- Inference times were measured on NVIDIA Tesla T4 and A100 GPUs, with the A100 showing faster performance. Due to its efficiency, the A100 was selected for offloading in HI and remote inference systems.
- Wi-Fi is preferred over BLE for image transmission due to significantly lower latency and energy consumption, making it more efficient for offloading tasks.
- The offloading time and energy consumption over Wi-Fi were measured by transmitting images and receiving inference results from the server, with results averaged over 10,000 images.

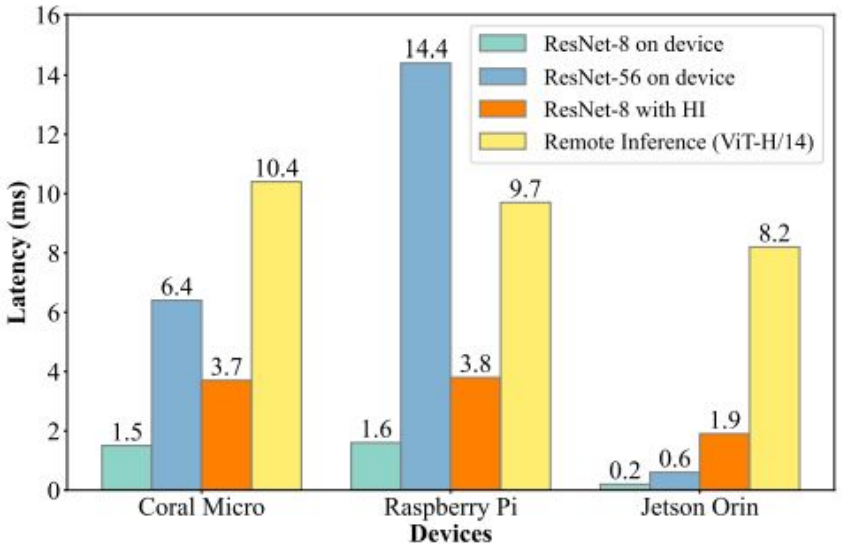
On-Device vs. HI vs. Remote Inference



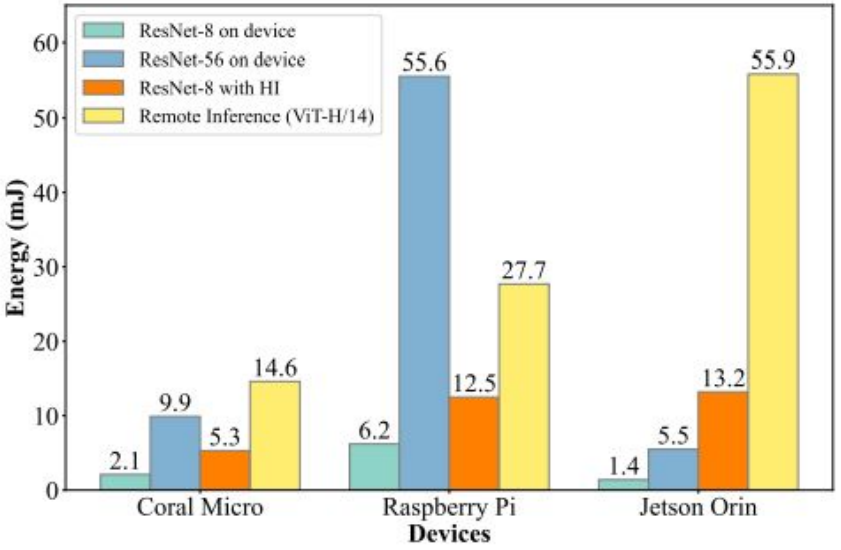
Comparison

		CIFAR-10			ImageNet-1K		
		ResNet-8	ResNet-56	AlexNet	ResNet-18	ResNet-50	AlexNet
Accuracy (%)	Coral	86.98	93.66	74.39	68.75	74.97	-
	Raspberry	86.97	93.53	74.27	69.67	75.86	56.36
	Jetson 15W	86.91	93.72	74.74	69.59	76.01	56.45
	Jetson 7W	86.91	93.72	74.74	69.59	76.01	56.45
Latency (ms)	Coral	1.50	6.45	69.41	168.22	690.43	-
	Raspberry	1.63	14.36	4.75	227.66	461.84	105.04
	Jetson 15W	0.17	0.64	0.39	0.84	1.80	1.48
	Jetson 7W	0.25	1.03	0.63	1.85	4.04	2.64
Energy (mJ)	Coral	2.14	9.86	112.00	230.68	939.69	-
	Raspberry	6.22	55.59	19.38	977.33	2004.21	446.08
	Jetson 15W	1.38	5.47	4.43	10.56	23.60	22.14
	Jetson 7W	1.72	7.12	5.59	16.13	35.71	27.90

Comparison CIFAR-10

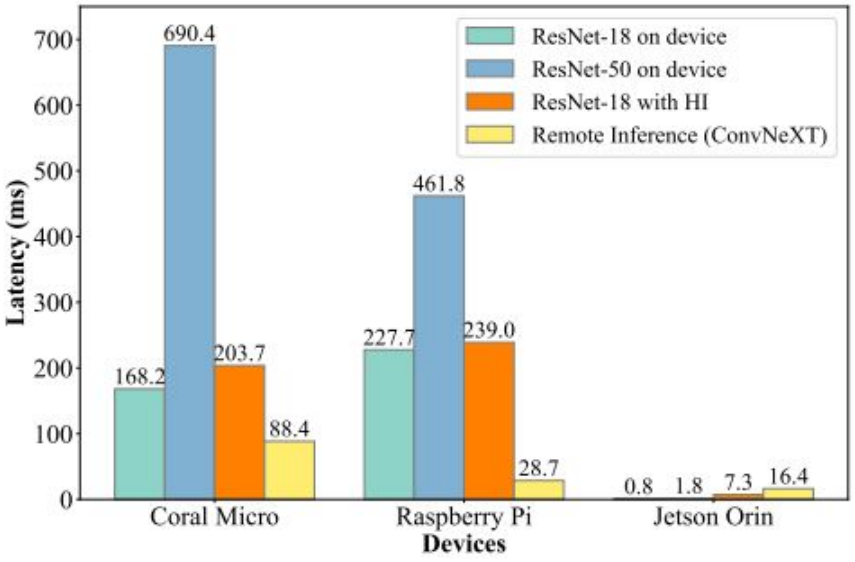


(a) Latency for CIFAR-10

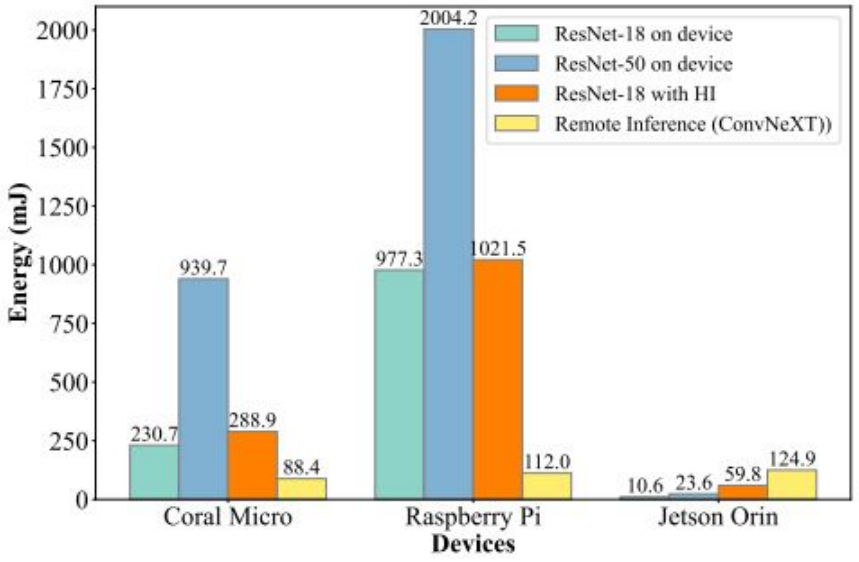


(b) Energy consumption for CIFAR-10

Comparison ImageNET-1k



(a) Latency for ImageNet-1K



(b) Energy consumption for ImageNet-1K

Final Results

- **Comparison:** Evaluated different strategies (on-device, HI, remote) for CIFAR-10 and ImageNet-1K on Coral Micro, Raspberry Pi, and Jetson Orin.
- **Key Insight:** Smaller models like ResNet-8 (CIFAR-10) and ResNet-18 (ImageNet-1K) offer lower latency and energy but don’t meet accuracy QoS. Larger models or remote inference are needed for higher accuracy.

- **CIFAR-10:** HI with ResNet-8 meets 90% accuracy QoS with up to 73% lower latency and 77% lower energy compared to on-device ResNet-56.
- **ImageNet-1K:** HI with ResNet-18 achieves better accuracy and efficiency on Jetson Orin compared to other on-device models.
- **For Complex Tasks:** Remote inference is generally more efficient on Coral Micro and Raspberry Pi, meeting accuracy and energy QoS for ImageNet-1K.
- **Jetson Orin:** On-device inference with RegNetY32GF is effective but HI still offers better accuracy and lower energy consumption.

Early Exit with Hierarchical Inference (EE-HI)

Early Exit (EE-HI) Strategy: Introduces early exit points in the model, reducing on-device overhead by allowing samples to exit early if confidence is high enough, minimizing unnecessary computations.

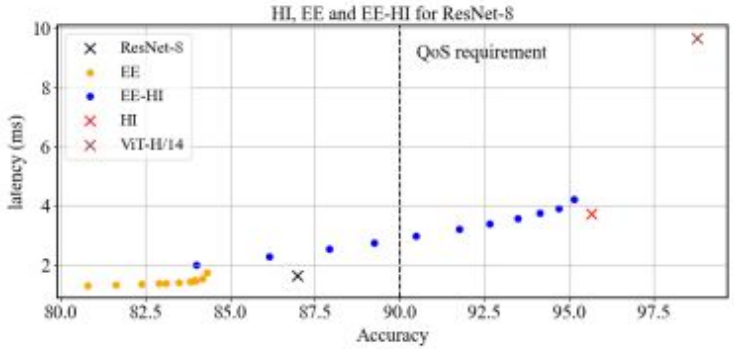
Model Design: EE branches are added at strategic points in models like ResNet-8, ResNet-56, and AlexNet, determined empirically to balance overhead and latency.

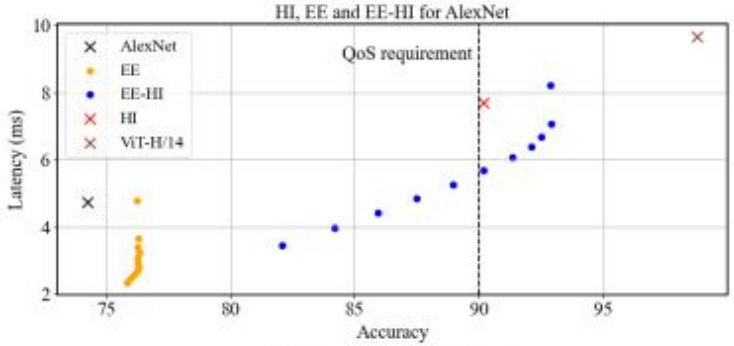
Implementation: The EE-HI system is designed based on the BranchyNet approach, enhancing HI by optimizing where and when samples are processed or offloaded.

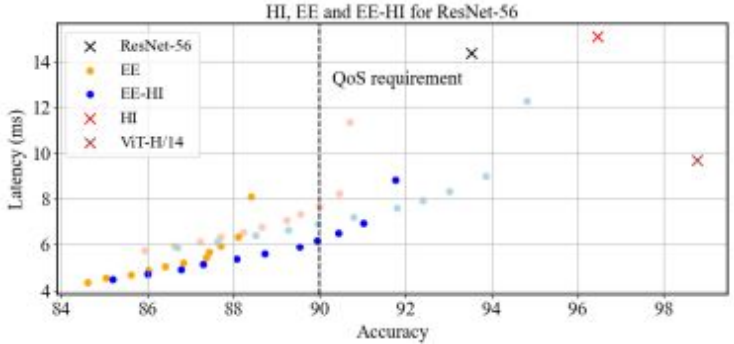
Model	Device	Pow.	EE-HI Time (ms)	EE-HI Energy (mJ)	HI Time (ms)	HI Energy (mJ)	EE+HI Threshold
ResNet-8	RaspPi		2.71	9.00	3.82	12.55	$\theta = 0.69$
	Jetson	7W	1.57	9.70	2.04	12.57	
		15W	1.47	10.26	1.87	13.05	
ResNet-56	RaspPi		6.13	23.02	15.21	58.10	$\theta_1 = 0.81$ $\theta_2 = 0.84$
	Jetson	7W	0.84	5.52	1.67	10.95	
		15W	0.69	5.22	1.24	9.63	
AlexNet	RaspPi		5.66	19.47	7.81	28.22	$\theta = 0.75$
	Jetson	7W	2.69	17.12	3.27	21.59	
		15W	2.51	17.98	2.90	21.53	

Optimization with EE-HI: By setting accuracy, latency, or energy constraints, optimal early exit thresholds are determined for EE-HI, significantly reducing latency and energy consumption (up to 60%) compared to the base HI strategy on devices like Raspberry Pi.

Trade-offs and Flexibility: EE-HI outperforms on-device-only models, offering a better accuracy-latency trade-off, especially in models like ResNet-56, making it the most effective strategy for meeting QoS requirements.







Summary of Preferred Strategies

- **CIFAR-10:** HI or EE-HI is preferred on most devices except Jetson Orin, where on-device ResNet-56 is optimal.
- **ImageNet-1K:** Remote inference is generally preferred except on Jetson Orin, where EE-HI with ResNet-18 is effective.

- **For Resource-Constrained Devices:** Offload complex tasks like CIFAR-10 (on Arduino Nano, ESP32) and ImageNet-1K (on Coral Micro, Raspberry Pi) to remote servers.
- **Optimal Strategy:** Use HI with smaller models when on-device inference is feasible; it balances accuracy, latency, and energy better than larger on-device models.