XiYOLO: Energy-Aware Object Detection via Iterative Architecture Search and Scaling

Tony Tran
Department of Research Computing
University of Houston
Houston, TX 77479
thtran37@cougarnet.uh.edu

Richie R. Suganda
Department of Electrical and Computer Engineering
University of Houston
Houston, TX 77479
rrsugand@cougarnet.uh.edu

Bin Hu
Department of Engineering Technology
University of Houston
Houston, TX 77479
bhu11@central.uh.edu

Abstract

Object detection on heterogeneous edge devices must satisfy strict energy, latency, and memory constraints while still providing reliable perception for downstream autonomy. Existing energy-aware NAS methods often target limited deployment settings, while real energy remains difficult to optimize because it is highly device-dependent and costly to measure. We address these challenges with an energy-adaptive framework that combines an energy-aware XiResOFA search space, a two-stage energy estimator, and iterative search to identify a single energy-efficient base architecture. We then apply compound scaling to transform this base design into the **XiYOLO** family across deployment budgets, enabling interpretable accuracy–energy tradeoffs under sparse hardware measurements. Experiments on PascalVOC, COCO, and real-device deployment show that XiYOLO achieves a stronger energy–accuracy tradeoff than YOLO baselines. On PascalVOC, the medium XiYOLO model reaches **86.15 mAP50** while reducing energy relative to YOLOv12m by **20.6%** on GPU and **35.9%** on NPU. On COCO, XiYOLO reduces energy relative to YOLOv12 by up to **53.7%** on GPU and **51.6%** on NPU at the small scale. The proposed two-stage estimator also improves sample efficiency over a joint predictor under few-shot adaptation with only **2–20** target-device samples.

1 Introduction

Object detection is increasingly important for autonomous systems deployed on heterogeneous edge devices, including UAVs, UGVs, mobile robots, and embedded cameras. These systems support tasks such as search and rescue, infrastructure inspection, and remote monitoring, where perception must run directly on-device under tight latency, memory, compute, and energy constraints [21, 4, 3, 26]. Among these constraints, energy is especially important because it directly affects system endurance. Larger detectors often improve accuracy, but they also increase energy and latency, whereas smaller models are more efficient but may sacrifice robustness and small-object sensitivity [31, 38, 35, 1]. As a result, the best detector is rarely universal across platforms or operating conditions, and edge perception should be treated as a *budget-adaptive* problem.

Neural architecture search (NAS) offers a natural framework for discovering detector operating points along this accuracy–energy frontier [6, 30, 9, 13, 36, 29], but applying NAS to energy-adaptive

Preprint.

edge perception remains difficult. Many existing methods still emphasize a single final model or a limited set of hardware settings rather than scalable detector families across heterogeneous devices [38, 15, 8, 34]. More importantly, energy is hard to optimize directly: FLOPs, MACs, and parameter count often fail to predict measured energy because energy also depends on memory movement, operator composition, and platform-specific execution behavior [5, 14, 39]. The central challenge is therefore not only how to search for efficient detectors, but how to make energy-aware detector design transferable across heterogeneous platforms when accurate hardware energy measurements are scarce.

In this paper, we address three key challenges in energy-adaptive object detection for heterogeneous edge devices: designing a search space with meaningful accuracy–energy tradeoffs, searching that space efficiently, and enabling energy-aware search when hardware energy data is scarce. To address these challenges, we design a novel XiResOFA search block, use an iterative search strategy over the backbone, FPN, and PAN, and develop a two-stage energy approximation model with a generic predictor and a hardware-specific residual. Together, these components identify a single energy-efficient base architecture, which we scale into a family of deployable detectors spanning different accuracy–energy operating regimes. In this way, the proposed framework treats edge detector design not as the search for a single efficient model, but as the search for budget-adaptive operating points aligned with heterogeneous hardware constraints.

Our contributions are summarized as follows:

- We formulate energy-adaptive perception as the design of an energy-efficient base detector architecture that can be scaled across deployment budgets on heterogeneous edge devices, and we instantiate this idea with XiYOLO, whose medium PascalVOC model achieves the highest **86.15 mAP50** among all compared medium-sized detectors.
- We propose an iterative neural architecture search framework for object detection that progressively searches the backbone, FPN, and PAN to identify an energy-efficient base architecture. After scaling, XiYOLO achieves strong accuracy–energy tradeoffs on both PascalVOC and COCO: on PascalVOC, the medium model improves over YOLOv12m by **+0.07 mAP50** while reducing energy by **20.6%** on GPU and **35.9%** on NPU; on COCO, the medium model reduces energy relative to YOLOv12m by **24.5%** on GPU and **38.8%** on NPU while maintaining competitive accuracy.
- We introduce an energy-aware search space based on a novel XiResOFA block with controllable compression ratio, kernel size, and lite/full attention choices, enabling detector search over interpretable accuracy–energy tradeoffs. Randomly sampled XiResOFA medium models reduce average energy by **21.1%** relative to the YOLOv12 medium baseline, while the iterative search space yields about **3.5%** higher average predicted mAP50 than the global search space.
- We develop a two-stage energy approximation model that combines a generic architecture–device energy prior with a lightweight hardware-specific residual model, reducing the amount of measured energy data required for energy-aware search, and we validate its sample efficiency on HW-NAS-Bench under few-shot adaptation with only **2–20** target-device samples.

2 Related Work

Efficient object detection for onboard deployment. Modern object detection has evolved from early real-time one-stage detectors such as SSD and YOLO to stronger multiscale architectures such as FPN, FCOS, and EfficientDet [24, 28, 22, 33, 31]. For resource-constrained deployment, prior work has developed lightweight detector families including MobileDets, TinyissimoYOLO, and FemtoDet, highlighting the need to balance detection accuracy against runtime and model cost on edge hardware [38, 25, 35]. In aerial and onboard settings, recent efforts have further emphasized efficient detection under limited compute and energy budgets [40, 20, 27, 12]. These works establish the importance of efficient onboard perception, but they typically target a single deployed detector rather than a set of models spanning different energy budgets.

Hardware-aware neural architecture search for detection. Neural architecture search has become an effective tool for automating efficient model design, and hardware-aware variants incorporate

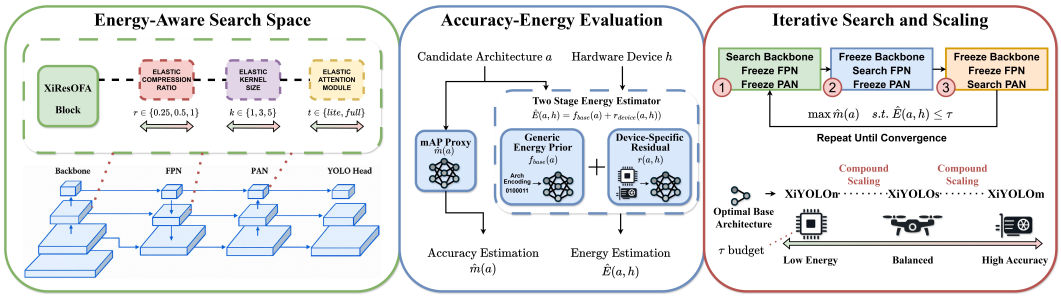


Figure 1: **Energy-aware NAS framework.** Candidate detectors from a searchable energy-aware architecture space are ranked by an mAP proxy and a two-stage energy estimator, then refined iteratively to obtain scalable models under hardware-specific energy constraints.

deployment-dependent costs such as latency or memory into the search objective [6, 30, 37]. In object detection, prior work has searched detector backbones, feature pyramids, and full detector pipelines, including DetNAS, NAS-FPN, NAS-FCOS, MobileDets, and DetOFA [9, 13, 36, 38, 29]. More recent work has moved toward edge- and hardware-aware detector NAS [15, 8, 34]. However, these approaches generally optimize for a single final architecture or a single deployment target. In contrast, our setting requires a *budget-adaptive detector family* that spans multiple operating points along the accuracy–energy frontier for onboard use.

Energy modeling for neural networks. Practical energy-aware search depends on being able to estimate hardware energy efficiently. Early work such as NeuralPower showed that neural network energy can be predicted from architectural characteristics, while more recent approaches have incorporated both network structure and device information for improved modeling [5, 14, 39]. At the same time, these studies reinforce that energy cannot be reliably reduced to FLOPs, MACs, or parameter count alone, since measured energy is strongly influenced by memory movement, operator composition, and device-specific execution behavior [5, 14]. This limitation is particularly important for detector NAS, where dense hardware measurements across a large search space are costly to obtain. Our work builds on this observation by coupling detector search with a two-stage energy approximation strategy designed for sparse hardware measurements.

3 Methodology

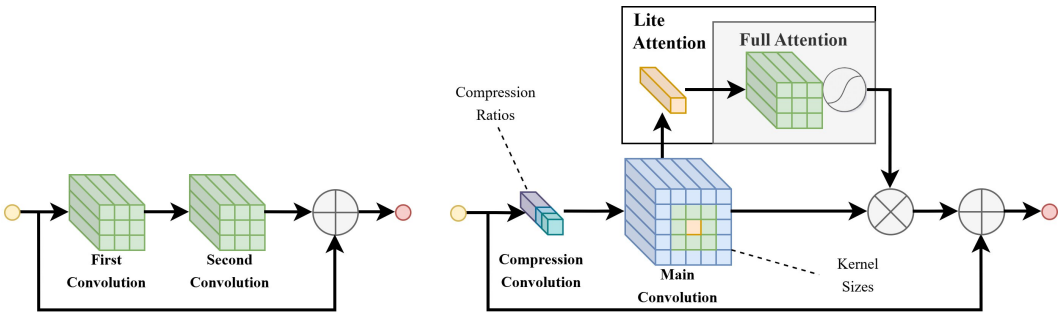
3.1 Problem Formulation and Framework Overview

We address onboard object detection under realistic robotic deployment constraints, where a perception model must operate within limited energy, latency, and memory budgets on a target hardware platform [21, 4, 3, 26]. Let h denote the target hardware platform and let $a \in \mathcal{A}$ denote a candidate detector architecture drawn from search space $\mathcal{A}$. We denote by $\hat{m}(a)$ the predicted detection accuracy of architecture a, measured through an mAP proxy, and by $\hat{E}(a, h)$ its estimated energy on hardware h. Our goal is to identify a single energy-efficient base detector that maximizes detection quality while satisfying a target energy budget. We therefore formulate architecture search as

$$a^\star = \arg\max_{a \in \mathcal{A}} \hat{m}(a) \quad \text{s.t.} \quad \hat{E}(a, h) \leq \tau,$$

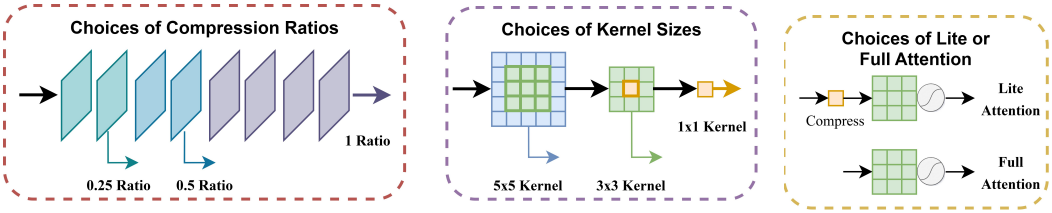
where τ is the deployment energy budget for hardware platform h.

Rather than directly searching for multiple detectors, the framework first identifies a single optimized base architecture and then derives deployment variants by scaling that architecture to different budget levels. This separates architecture discovery from deployment adaptation: iterative search finds the base design, while post-search scaling produces high-accuracy, balanced, and low-energy variants [7, 29, 34]. As shown in Figure 1, candidate detectors are sampled from an energy-aware search space over the backbone, FPN, and PAN, then evaluated using a detection-quality proxy and a two-stage energy estimator that combines a generic architecture predictor with a lightweight device-specific correction term [5, 14, 39]. These signals guide an iterative NAS loop toward a single energy-efficient



(a) Default Bottleneck Block (b) XiResOFA Block

Figure 2: Comparison between (a) the standard bottleneck block [16] and (b) the proposed XiResOFA block. XiResOFA augments a residual design with elastic architectural choices for energy-aware detector search.



(a) Compression Ratio (b) Kernel Size (c) Lite/Full Attention

Figure 3: Elastic architectural choices in XiResOFA: selectable (a) compression ratios, (b) kernel sizes, and (c) lite or full attention mechanisms.

operating point, after which the searched base architecture is scaled to match different resource budgets [7].

3.2 Energy-Aware Search Space

A central design requirement of the search space is that its architectural choices should have a direct and interpretable relationship to both detection quality and hardware energy. Figure 2 compares the standard residual bottleneck block with the proposed *XiResOFA* block. The standard bottleneck provides stable optimization through identity shortcuts [16], but it does not directly expose elastic operator-level choices suitable for energy-aware detector search. In contrast, XiResOFA preserves a residual shortcut while introducing configurable architectural choices compatible with once-for-all search [7]. This design makes the block both trainable and searchable, while allowing its internal structure to adapt to different accuracy–energy operating points.

Figure 3 illustrates the three elastic choices exposed by each XiResOFA block. Each block is parameterized by a tuple $b = (r, k, t)$, where $r \in \{1, 0.5, 0.25\}$ is the compression ratio, $k \in \{1, 3, 5\}$ is the kernel size, and $t \in \{\text{lite}, \text{full}\}$ is the attention type. These dimensions induce clear accuracy–energy tradeoffs. Lower compression preserves more channels and typically improves representational strength, but increases arithmetic cost and activation movement. Larger kernels expand the receptive field and may improve contextual reasoning, especially for multiscale detection, but also increase energy consumption. Full attention provides stronger feature recalibration than a lightweight alternative, but introduces additional compute and memory overhead. By embedding these choices within a shared supernet, the search space can represent a range of candidate architectures from aggressively efficient configurations to higher-capacity detectors [7, 30, 37].

We apply XiResOFA consistently across the major stages of a one-stage detector, including the backbone, FPN, and PAN. The backbone extracts hierarchical feature representations, the FPN performs multiscale semantic fusion, and the PAN strengthens bottom-up aggregation for localization and small-object reasoning [22, 23]. A candidate detector is therefore written as $a = \left(a^{\text{back}}, a^{\text{fpn}}, a^{\text{pan}}\right)$, where each component denotes a sequence of XiResOFA block choices. The full search space is

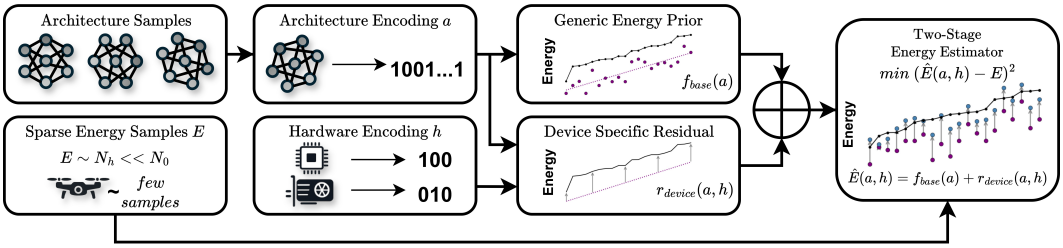


Figure 4: Overview of the proposed two-stage energy estimator. Sparse energy samples from the target hardware are combined with architecture and hardware encodings to learn a generic energy prior and a device-specific residual.

$\mathcal{A} = \mathcal{A}_{\text{back}} \times \mathcal{A}_{\text{fpn}} \times \mathcal{A}_{\text{pan}}$. Using the same searchable block family across all three stages provides a coherent detector-level search space rather than imposing unrelated search rules across different modules. It also enables efficient weight sharing across subnetworks, which substantially reduces the cost of exploring detector candidates during search [7, 29].

3.3 Two-Stage Energy Estimation

A major obstacle in energy-aware detector search is that true hardware energy is expensive to measure at scale [5, 14, 39]. Simple analytical surrogates such as FLOPs, MACs, or parameter count are often insufficient because measured energy also depends on memory traffic, intermediate activations, operator composition, and hardware-specific execution behavior [5, 14]. For object detectors, this makes direct energy optimization particularly difficult, since exhaustive profiling over a large search space is costly and often impractical.

Figure 4 illustrates the proposed two-stage energy estimation pipeline. Let N_0 denote the full number of candidate architectures in the search space and let N_h denote the number of measured energy samples available on target hardware h, where in practice $N_h \ll N_0$. Given architecture samples and sparse energy measurements from the target device, we encode both the detector architecture and the hardware target into compact representations and decompose energy prediction into a generic prior and a device-specific correction. For a candidate architecture a on hardware h, we estimate energy as

$$\hat{E}(a, h) = f_{\text{base}}(a) + r(a, h),$$

where $f_{\text{base}}(a)$ is a generic pretrained energy predictor and $r(a, h)$ is a lightweight hardware-specific residual model. The first stage captures transferable architecture-level energy structure, while the second stage adapts that prior to device-specific effects such as backend implementation and memory hierarchy [14, 39]. Because the residual only models the remaining hardware-specific discrepancy, it can be calibrated with relatively few measured samples, that is, using only N_h target-device measurements instead of exhaustively profiling all N_0 candidates. During search, this two-stage predictor replaces direct hardware measurement and enables practical hardware-aware ranking of detector candidates under sparse energy supervision.

3.4 Energy-Aware Iterative Search and Scaling

Jointly searching over the backbone, FPN, and PAN creates a large combinatorial design space and can make hardware-aware detector NAS expensive and difficult to optimize [9, 13, 36, 29]. These components also influence accuracy and energy differently: backbone choices largely determine representational strength and compute load, while FPN and PAN choices affect multiscale fusion, feature reuse, and memory traffic [22, 23]. To exploit this structure, we use an iterative search procedure that progressively refines detector components rather than performing a single monolithic joint search.

At iteration t, let the current detector be $a_t = \left(a_t^{\text{back}}, a_t^{\text{fpn}}, a_t^{\text{pan}}\right)$. We alternately optimize each detector stage while conditioning on the current choices of the others:

$$a_{t+1}^{\text{back}} = \arg\max_{b \in \mathcal{A}_{\text{back}}} \hat{m}\Big(b, a_t^{\text{fpn}}, a_t^{\text{pan}}\Big) \quad \text{s.t.} \quad \hat{E}\Big(b, a_t^{\text{fpn}}, a_t^{\text{pan}}, h\Big) \le \tau,$$

$$a_{t+1}^{\text{fpn}} = \arg\max_{f \in \mathcal{A}_{\text{fpn}}} \hat{m}\big(a_{t+1}^{\text{back}}, f, a_t^{\text{pan}}\big) \quad \text{s.t.} \quad \hat{E}\big(a_{t+1}^{\text{back}}, f, a_t^{\text{pan}}, h\big) \le \tau,$$

$$a_{t+1}^{\text{pan}} = \arg\max_{p \in \mathcal{A}_{\text{pan}}} \hat{m}\Big(a_{t+1}^{\text{back}}, a_{t+1}^{\text{fpn}}, p\Big) \quad \text{s.t.} \quad \hat{E}\Big(a_{t+1}^{\text{back}}, a_{t+1}^{\text{fpn}}, p, h\Big) \le \tau.$$

Each step selects the highest-scoring candidate that satisfies the energy budget, allowing the backbone and neck to co-adapt over successive iterations. This stage-wise refinement reduces search complexity while preserving the main accuracy–energy tradeoff. After T iterations, the final searched base architecture is $a^\star = a_T$.

After search, we derive a detector family by scaling the selected base architecture to different deployment budgets. Let $S(a^\star, \alpha)$ denote a scaling operator with scale factor α. We obtain the final deployment variants as

$$a^{(\ell)} = S(a^\star, \alpha_\ell), \qquad \ell \in \{n, s, m\},$$

where n, s, and m denote nano, small, and medium variants. The searched model serves as the balanced reference design, while compound scaling adjusts width, depth, or related stage capacity to produce higher- and lower-budget variants [7]. In this way, iterative NAS identifies the core detector design, and post-search scaling adapts it to different operating regimes without requiring separate searches for each deployment point.

4 Experiments

4.1 Experimental Setup

Search space. We construct the detector search space by starting from the base YOLOv12n architecture [32] and replacing each bottleneck block with the proposed XiResidualOFA block. Each XiResidualOFA block exposes three choices: compression ratio, kernel size, and attention type. For compression, we use ratios of $\{0.25, 0.5, 1\}$, where smaller ratios preserve more channels and generally improve accuracy at higher energy cost, while larger ratios are more efficient but less expressive. For kernel size, we use $\{1, 3, 5\}$, where larger kernels provide stronger spatial context at higher energy cost. For attention, we consider *full* and *lite* variants, where full attention is more expressive but more expensive. The resulting detector search space is partitioned into a backbone space with 18^2 configurations, an FPN space with 18^4 configurations, and a PAN space with 18^4 configurations.

Training protocol. On PascalVOC [11], we train the supernet for 300 epochs on one RTX 4090 GPU with batch size 64 using the largest supernet configuration, followed by 300 epochs of uniform subnet training by randomly sampling one subnet per batch. We then sample 1000 subnetworks to train an mAP proxy and an energy proxy, both implemented as MLPs with hidden size 400, and perform iterative search for 4 iterations over the backbone, FPN, and PAN to obtain a single optimized base architecture. The searched subnet is fine-tuned for 300 epochs, then compound scaled to small and medium variants, which are also fine-tuned for 300 epochs. On COCO, we follow the same procedure and search space, but train using 8 V100 GPUs. The searched architectures for PascalVOC and COCO are obtained independently. We refer to the resulting detector family as **XiYolo**.

Deployment platforms and baselines. We evaluate the searched models against YOLO baselines on the ModalAI Sentinel Development Drone, which contains a Qualcomm QRB5165 CPU, a Qualcomm Adreno 650 GPU, and a 15 TOPS NPU. We compare against YOLOv5 [17], YOLOv8 [18], YOLO11 [18], and YOLOv12 [32] at nano, small, and medium scales. All models are exported to FP16 TFLite, and we measure energy per inference, cumulative energy over time, latency, and detection accuracy. Power is monitored using the VOXL Power Module v3. We estimate inference energy by subtracting idle power from measured power to obtain active inference power, then multiplying by inference time and aggregating over 1000 samples.

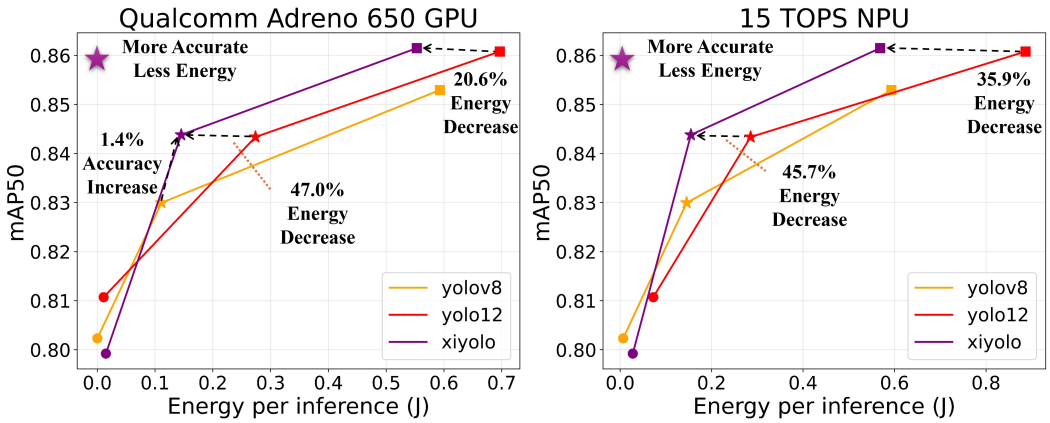


Figure 5: Comparisons with other YOLO-series [18, 32] on PascalVOC dataset in terms of energy and accuracy on the GPU (left) and NPU (right).

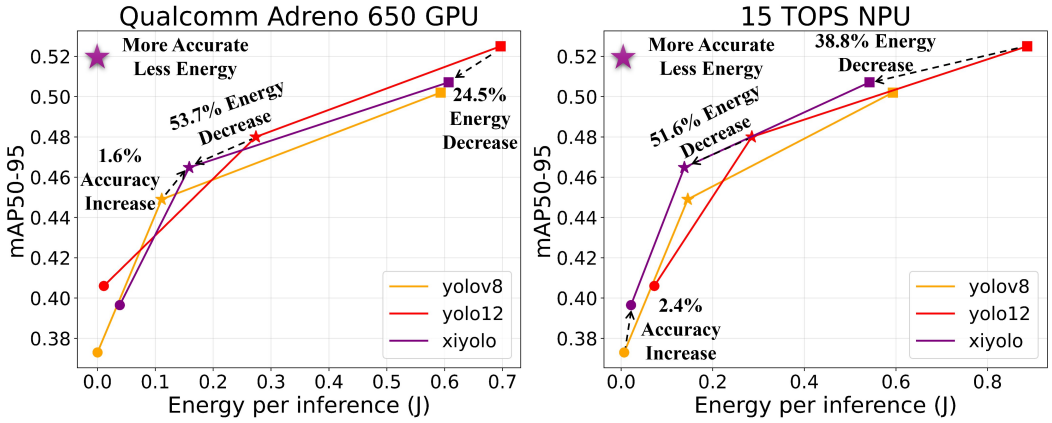


Figure 6: Comparisons with other YOLO-series [18, 32] on COCO dataset in terms of energy and accuracy on the GPU (left) and NPU (right).

4.2 Accuracy and Energy Tradeoffs

We first evaluate the deployment quality of the searched models by comparing detection accuracy and energy jointly. Figures 5 and 6 compare XiYOLO against YOLOv8 and YOLOv12 on PascalVOC and COCO, respectively, on the Qualcomm Adreno 650 GPU and the 15 TOPS NPU. Since each model is evaluated from a fixed trained checkpoint, accuracy remains constant across devices, while energy changes with the deployment backend. The key result is that XiYOLO consistently improves the accuracy–energy tradeoff relative to YOLOv12, with the clearest gains at the small and medium scales.

On PascalVOC, XiYOLO achieves the strongest overall tradeoff on both GPU and NPU. At the medium scale, XiYOLO reaches the highest **86.15 mAP50**, improving over YOLOv12m by **+0.07 mAP50**, while reducing energy by **20.6%** on GPU and **35.9%** on NPU. At the small scale, XiYOLO reaches **84.38 mAP50**, slightly above YOLOv12s at **84.34 mAP50**, while reducing energy by **47.0%** on GPU and **45.7%** on NPU. Relative to YOLOv8, XiYOLO also provides clear accuracy gains, including **+1.4%** mAP50 at the small scale. These results show that on PascalVOC, XiYOLO improves accuracy while simultaneously lowering energy, especially in the practically relevant small and medium operating regimes.

On COCO, XiYOLO again shows a favorable accuracy–energy tradeoff, although the gains are more selective. At the medium scale, XiYOLO reduces energy relative to YOLOv12m by **24.5%** on GPU and **38.8%** on NPU, while maintaining competitive accuracy (**50.71** vs. **52.50 mAP50-95**). At the small scale, XiYOLO improves over YOLOv8s by **+1.6%** mAP50-95 on GPU and over

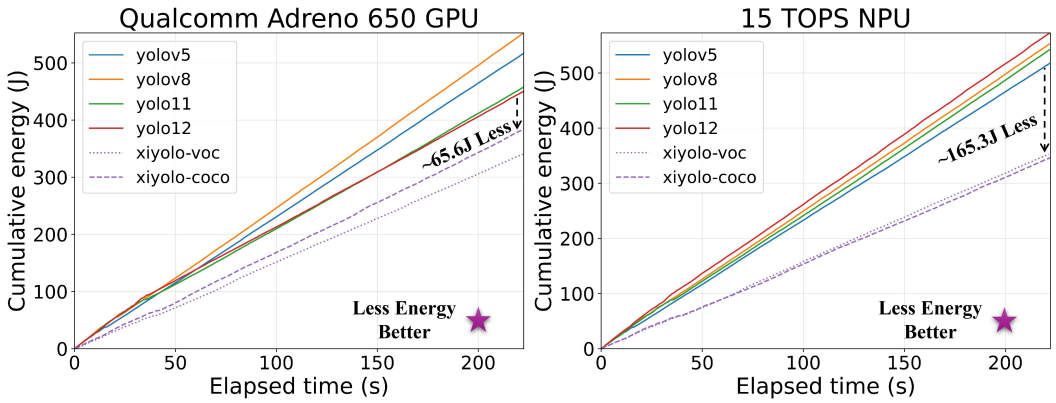


Figure 7: Energy consumption vs. time benchmarks against other YOLO-series [17, 18, 32] on the GPU (left) and NPU (right) for the medium model scales.

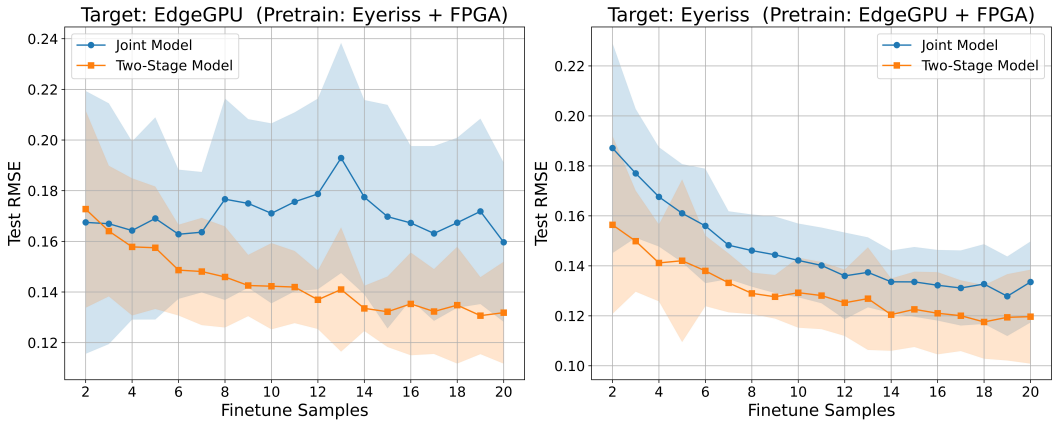


Figure 8: Two-Stage Energy Estimator vs. Joint Model under few-shot adaptation with 2–20 samples on HW-NAS-Bench [19], using the NAS-Bench-201 [10] search space and the EdgeGPU and Eyeriss hardware targets.

YOLOv12s by **+2.4%** mAP50-95 on NPU, while also reducing energy relative to YOLOv12s by **53.7%** on GPU and **51.6%** on NPU. Overall, Figures 5 and 6 show that XiYOLO consistently shifts the accuracy–energy frontier in a favorable direction, with especially strong gains against YOLOv12 and the most consistent benefits appearing at the small and medium scales.

4.3 Deployment Energy and Scaling Behavior

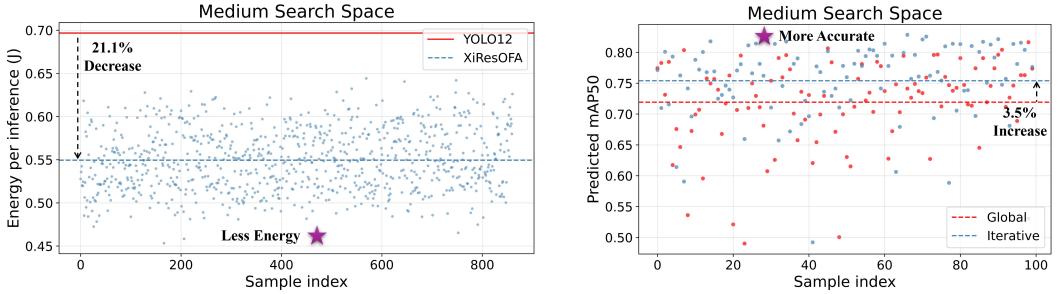
We next examine how the searched architectures behave during sustained deployment. Figure 7 reports cumulative energy over time for medium-scale models on the Qualcomm Adreno 650 GPU and the 15 TOPS NPU. The clearest result is that both **XiYOLO-VOC** and **XiYOLO-COCO** accumulate substantially less energy over time than the YOLO baselines on both devices. This shows that the searched architectures preserve their efficiency advantage not only in per-inference measurements, but also under continuous deployment.

On the GPU, both XiYOLO variants remain below all YOLO baselines throughout the benchmark horizon, with **XiYOLO-VOC** achieving the lowest cumulative energy overall. By the end of the run, XiYOLO-COCO uses roughly **65.6 J** less cumulative energy than the strongest YOLO baseline. On the NPU, the separation is even larger: both XiYOLO variants again remain below all YOLO baselines, and XiYOLO-COCO achieves the lowest cumulative energy, using about **165.3 J** less cumulative energy than the strongest YOLO baseline. Overall, Figure 7 shows that the searched XiYOLO architectures deliver the best deployment-time energy behavior among all compared medium-sized models on both accelerator backends.

4.4 Energy Estimation with Sparse Hardware Samples

We next evaluate the proposed two-stage energy estimator under limited device-specific supervision. Figure 8 reports results on HW-NAS-Bench, using the NAS-Bench-201 search space and two hardware targets: EdgeGPU and Eyeriss. For each target device, we pretrain the estimator on the remaining hardware targets and then fine-tune on the target device using only 2 to 20 samples. We compare the proposed two-stage estimator against a joint model under the same training protocol and report test RMSE on the held-out target device [19, 10, 14, 39].

Figure 8 shows that the two-stage estimator achieves lower test RMSE than the joint model on both EdgeGPU and Eyeriss across nearly the full few-shot range. The improvement is visible even in the lowest-data regime and remains consistent as more target-device samples are added. These results indicate that separating generic architecture-level energy structure from hardware-specific residual correction yields more sample-efficient adaptation than directly fitting a single joint predictor [14, 39].



(a) Energy distribution of randomly sampled XiResOFA medium models.

(b) Predicted accuracy of sampled models from global and iterative search spaces.

Figure 9: Energy and accuracy characteristics of the medium search space.

4.5 Search Space Characteristics

We next examine the properties of the proposed medium-scale search space in terms of both energy and accuracy. Figure 9(a) shows the energy distribution of randomly sampled XiResOFA models. Compared with the YOLOv12 medium baseline, the sampled models consistently occupy a lower-energy region, with the average energy reduced by **21.1%**. This indicates that the proposed search space is biased toward more energy-efficient designs while still spanning a broad set of candidate architectures.

Figure 9(b) compares the predicted accuracy of sampled architectures from the global and iterative search spaces. The iterative search space yields a higher average predicted mAP50 than the global search space, with an improvement of about **3.5%**. This suggests that the iterative decomposition produces a more favorable candidate region for optimization, concentrating search on architectures with stronger expected accuracy. Together, these results show that the proposed XiResOFA design space is both energy-efficient and structurally well suited for iterative search.

5 Conclusion

We presented an energy-adaptive framework for object detection on heterogeneous edge devices that combines an energy-aware search space, a two-stage energy estimator, and iterative neural architecture search. The proposed XiResOFA search space exposes architectural choices that directly control the accuracy–energy tradeoff, while the two-stage estimator enables energy-aware search under sparse hardware measurements. By searching for a single energy-efficient base architecture and scaling it into the XiYOLO family, our framework achieves strong accuracy–energy tradeoffs across GPU and NPU deployment. On PascalVOC, XiYOLO improves over YOLOv12 at the small and medium scales while substantially reducing energy on both devices. On COCO, XiYOLO again delivers large energy reductions relative to YOLOv12, especially at the small scale, while maintaining competitive medium-scale accuracy. We also show that XiYOLO achieves the best deployment-time

energy behavior among compared medium-sized models and that the proposed two-stage estimator improves few-shot adaptation over a joint predictor on HW-NAS-Bench.

This work has both positive and negative societal implications. More energy-efficient edge perception can improve the practicality of search and rescue, disaster response, infrastructure inspection, and environmental monitoring by extending deployment time and reducing reliance on remote compute. At the same time, efficient onboard perception could also enable harmful uses such as persistent surveillance or reduced human oversight. Our study also has several limitations: the search is confined to a YOLOv12-derived detector family, the final model family is obtained through post-search scaling rather than direct multi-point search, and the evaluation focuses on selected GPU and NPU deployment settings. Despite these limitations, our results show that energy-aware architecture search can produce detector designs that are both accurate and energy-efficient for heterogeneous edge deployment.

References

[1] Daghash K. Alqahtani, Muhammad Aamir Cheema, and Adel N. Toosi. Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices. In Walid Gaaloul, Michael Sheng, Qi Yu, and Sami Yangui, editors, *Service-Oriented Computing*, pages 142–150, Singapore, 2025. Springer Nature. ISBN 978-981-96-0805-8. doi: 10.1007/978-981-96-0805-8_11.

[2] Alberto Ancilotto, Francesco Paissan, and Elisabetta Farella. XiNet: Efficient Neural Networks for tinyML. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16968–16977, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Ancilotto_XiNet_Efficient_Neural_Networks_for_tinyML_ICCV_2023_paper.html.

[3] Colby Banbury, Chuteng Zhou, Igor Fedorov, Ramon Matas, Urmish Thakker, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul Whatmough. MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers. *Proceedings of Machine Learning and Systems*, 3:517–532, March 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/hash/c4d41d9619462c534b7b61d1f772385e-Abstract.html.

[4] Alessio Burrello, Angelo Garofalo, Nazareno Bruschi, Giuseppe Tagliavini, Davide Rossi, and Francesco Conti. DORY: Automatic End-to-End Deployment of Real-World DNNs on Low-Cost IoT MCUs. *IEEE Transactions on Computers*, 70(8):1253–1268, August 2021. ISSN 1557-9956. doi: 10.1109/TC.2021.3066883. URL https://ieeexplore.ieee.org/document/9381618.

[5] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. NeuralPower: Predict and Deploy Energy-Efficient Convolutional Neural Networks. In *Proceedings of the Ninth Asian Conference on Machine Learning*, pages 622–637. PMLR, November 2017. URL https://proceedings.mlr.press/v77/cai17a.html.

[6] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *2018 International Conference on Learning Representations (ICLR)*, September 2018. URL https://openreview.net/forum?id=HylVB3AqYm.

[7] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for All: Train One Network and Specialize it for Efficient Deployment. In *2020 International Conference on Learning Representations (ICLR)*, April 2020. URL https://iclr.cc/virtual_2020/poster_HylxE1HKwS.html.

[8] Oscal Tzyh-Chiang Chen, Yu-Xuan Chang, Chih-Yu Chung, Ya-Yun Cheng, and Manh-Hung HA. Hardware-Aware Iterative One-Shot Neural Architecture Search With Adaptable Knowledge Distillation for Efficient Edge Computing. *IEEE Access*, 13:54204–54222, 2025. ISSN 2169-3536. doi: 10.1109/ACCESS.2025.3554185. URL https://ieeexplore.ieee.org/document/10938148/.

[9] Yukang Chen, Tong Yang, Xiangyu Zhang, GAOFENG MENG, Xinyu Xiao, and Jian Sun. DetNAS: Backbone Search for Object Detection. In *Advances*

in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/228b25587479f2fc7570428e8bcbabdc-Abstract.html.

[10] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *2019 International Conference on Learning Representations (ICLR)*, September 2019. URL https://openreview.net/forum?id=HJxyZkBKDr.

[11] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Int J Comput Vis*, 88 (2):303–338, June 2010. ISSN 1573-1405. doi: 10.1007/s11263-009-0275-4. URL https://doi.org/10.1007/s11263-009-0275-4.

[12] Wanxuan Geng, Junfan Yi, and Liang Cheng. An efficient detector for maritime search and rescue object based on unmanned aerial vehicle images. *Displays*, 87:102994, April 2025. ISSN 0141-9382. doi: 10.1016/j.displa.2025.102994. URL https://www.sciencedirect.com/science/article/pii/S0141938225000319.

[13] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7029–7038, June 2019. doi: 10.1109/CVPR.2019.00720. URL https://ieeexplore.ieee.org/document/8954436. ISSN: 2575-7075.

[14] Chaopeng Guo, Shiyu Wang, Ruolan Xie, and Jie Song. Estimating energy consumption of neural networks with joint Structure–Device encoding. *Sustainable Computing: Informatics and Systems*, 45:101062, January 2025. ISSN 2210-5379. doi: 10.1016/j.suscom.2024.101062. URL https://www.sciencedirect.com/science/article/pii/S2210537924001070.

[15] Diksha Gupta, Rhui Dih Lee, and Laura Wynter. On Efficient Object-Detection NAS for ADAS on Edge devices. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 1005–1010, June 2024. doi: 10.1109/CAI59869.2024.00183. URL https://ieeexplore.ieee.org/document/10605392.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90. URL https://ieeexplore.ieee.org/document/7780459. ISSN: 1063-6919.

[17] Glenn Jocher. YOLOv5 by Ultralytics, May 2020. URL https://github.com/ultralytics/yolov5.

[18] Glenn Jocher. YOLO by Ultralytics, May 2026. URL https://github.com/ultralytics/ultralytics.

[19] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan Lin. HW-NAS-BENCH: HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH BENCHMARK. In *2021 International Conference on Learning Representations (ICLR)*, 2021.

[20] Chan Yue Liew, Joanne Mun-Yee Lim, Chee Pin Tan, and Raja Mazhar Mohar Bin Tun Mohar. Altitude-informed fusion pyramid network for multi-scale waste detection in unmanned aerial vehicle images. *Engineering Applications of Artificial Intelligence*, 153: 110814, August 2025. ISSN 0952-1976. doi: 10.1016/j.engappai.2025.110814. URL https://www.sciencedirect.com/science/article/pii/S0952197625008140.

[21] Ji Lin, Wei-Ming Chen, Yujun Lin, john cohn, Chuang Gan, and Song Han. MCUNet: Tiny Deep Learning on IoT Devices. In *Advances in Neural Information Processing Systems*, volume 33, pages 11711–11722. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html.

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017. doi: 10.1109/CVPR.2017.106. URL https://ieeexplore.ieee.org/document/8099589. ISSN: 1063-6919.

[23] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path Aggregation Network for Instance Segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, June 2018. doi: 10.1109/CVPR.2018.00913. URL https://ieeexplore.ieee.org/document/8579011. ISSN: 2575-7075.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0. doi: 10.1007/978-3-319-46448-0_2.

[25] Julian Moosmann, Marco Giordano, Christian Vogt, and Michele Magno. TinyissimoYOLO: A Quantized, Low-Memory Footprint, TinyML Object Detection Network for Low Power Microcontrollers. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–5, June 2023. doi: 10.1109/AICAS57966.2023.10168657. URL https://ieeexplore.ieee.org/document/10168657. ISSN: 2834-9857.

[26] Arthur Moss, Hyunjong Lee, Lei Xun, Chulhong Min, Fahim Kawsar, and Alessandro Montanari. Ultra-Low Power DNN Accelerators for IoT: Resource Characterization of the MAX78000. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, pages 934–940, New York, NY, USA, January 2023. Association for Computing Machinery. ISBN 978-1-4503-9886-2. doi: 10.1145/3560905.3568300. URL https://dl.acm.org/doi/10.1145/3560905.3568300.

[27] Van Quang Nghiem, Huy Hoang Nguyen, and Minh Son Hoang. LEAF-YOLO: Lightweight Edge-Real-Time Small Object Detection on Aerial Imagery. *Intelligent Systems with Applications*, 25:200484, March 2025. ISSN 2667-3053. doi: 10.1016/j.iswa.2025.200484. URL https://www.sciencedirect.com/science/article/pii/S2667305325000109.

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016. doi: 10.1109/CVPR.2016.91. URL https://ieeexplore.ieee.org/document/7780460. ISSN: 1063-6919.

[29] Yuiko Sakuma, Masato Ishii, and Takuya Narihira. DetOFA: Efficient Training of Once-for-All Networks for Object Detection using Path Filter. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1325–1334, October 2023. doi: 10.1109/ICCVW60793.2023.00143. URL https://ieeexplore.ieee.org/document/10350829. ISSN: 2473-9944.

[30] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Tan_MnasNet_Platform-Aware_Neural_Architecture_Search_for_Mobile_CVPR_2019_paper.

[31] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, June 2020. doi: 10.1109/CVPR42600.2020.01079. URL https://ieeexplore.ieee.org/document/9156454. ISSN: 2575-7075.

[32] Yunjie Tian, Qixiang Ye, and David Doermann. YOLOv12: Attention-Centric Real-Time Object Detectors. In *Advances in Neural Information Processing Systems*, October 2025. URL https://openreview.net/forum?id=gCvByDI4FN.

[33] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: A Simple and Strong Anchor-Free Object Detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (4):1922–1933, April 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3032166. URL https://ieeexplore.ieee.org/document/9229517.

[34] Tony Tran, Qin Lin, and Bin Hu. ELASTIC: Efficient Once For All Iterative Search for Object Detection on Microcontrollers. *IEEE Transactions on Computers*, (01):1–8, March 2026. ISSN 0018-9340. doi: 10.1109/TC.2026.3678184. URL https://www.computer.org/csdl/journal/tc/5555/01/11456502/2faQWQg7unK.

[35] Peng Tu, Xu Xie, Guo Ai, Yuexiang Li, Yawen Huang, and Yefeng Zheng. FemtoDet: An Object Detection Baseline for Energy Versus Performance Tradeoffs. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13272–13281, October 2023. doi: 10.1109/ICCV51070.2023.01225. URL https://ieeexplore.ieee.org/document/10376762. ISSN: 2380-7504.

[36] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. NAS-FCOS: Fast Neural Architecture Search for Object Detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11940–11948, June 2020. doi: 10.1109/CVPR42600.2020.01196. URL https://ieeexplore.ieee.org/document/9156326. ISSN: 2575-7075.

[37] 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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10726–10734, June 2019. doi: 10.1109/CVPR.2019.01099. URL https://ieeexplore.ieee.org/document/8953587. ISSN: 2575-7075.

[38] Yunyang Xiong, Hanxiao Liu, Suyog Gupta, Berkin Akin, Gabriel Bender, Yongzhe Wang, Pieter-Jan Kindermans, Mingxing Tan, Vikas Singh, and Bo Chen. MobileDets: Searching for Object Detection Architectures for Mobile Accelerators. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3824–3833, June 2021. doi: 10.1109/CVPR46437.2021.00382. URL https://ieeexplore.ieee.org/document/9578555. ISSN: 2575-7075.

[39] Jiaru Zhang, Zesong Wang, Hao Wang, Tao Song, Huai-an Su, Rui Chen, Yang Hua, Xiangwei Zhou, Ruhui Ma, Miao Pan, and Haibing Guan. AMPERE: A Generic Energy Estimation Approach for On-Device Training. *SIGMETRICS Perform. Eval. Rev.*, 53(2):27–32, August 2025. ISSN 0163-5999. doi: 10.1145/3764944.3764951. URL https://dl.acm.org/doi/10.1145/3764944.3764951.

[40] Liming Zhou, Xiaohan Rao, Yahui Li, Xianyu Zuo, Yang Liu, Yinghao Lin, and Yong Yang. SWDet: Anchor-Based Object Detector for Solid Waste Detection in Aerial Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 16:306–320, 2023. ISSN 2151-1535. doi: 10.1109/JSTARS.2022.3218958. URL https://ieeexplore.ieee.org/document/9935119.

A Full Deployment Energy Results

Figure 10 reports the full cumulative-energy trajectories for all nano, small, and medium models on the Qualcomm Adreno 650 GPU and the 15 TOPS NPU. These results complement the main-text medium-scale comparison by showing the complete deployment behavior across all scales and baselines.

Several trends are consistent across the full benchmark. First, the strongest energy advantages of XiYOLO appear at the medium scale. On the GPU, **XiYOLOm** achieves the lowest cumulative energy among all compared medium-sized models at **340.2 J**, compared with **449.5 J** for YOLOv12m, **456.5 J** for YOLO11m, **516.6 J** for YOLOv5mu, and **552.4 J** for YOLOv8m. On the NPU, **XiYOLOm** again outperforms all YOLO baselines at **352.9 J**, compared with **572.6 J** for YOLOv12m. These results are consistent with the main-text finding that the searched XiYOLO architecture provides the strongest deployment-time energy behavior in the medium regime.

Second, the relative gains at the small and nano scales are more mixed, but XiYOLO still remains competitive and often improves substantially over YOLOv12. On the GPU, **XiYOLOs** reduces cumulative energy relative to YOLOv12s from **168.6 J** to **86.9 J**. On the NPU, the same comparison drops from **173.5 J** for YOLOv12s to **90.0 J** for **XiYOLOs**. At the nano scale, XiYOLO also reduces cumulative energy relative to YOLOv12n on both accelerators, lowering GPU energy from **6.17 J** to **5.45 J** and NPU energy from **50.45 J** to **15.23 J**. However, some other baselines remain more energy-efficient at these lower-capacity operating points, especially YOLOv5 and YOLO11 in selected nano and small settings. Overall, the full results show that XiYOLO maintains the clearest advantage at the medium scale, while remaining competitive and often substantially more efficient than YOLOv12 across the broader deployment space.

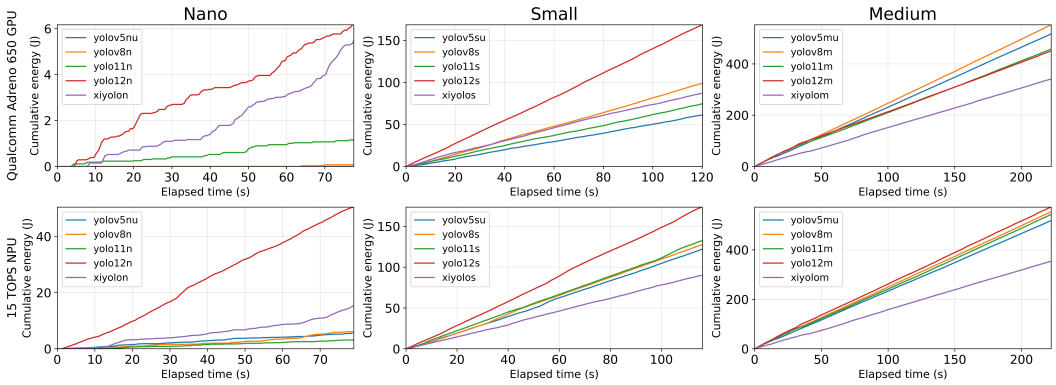


Figure 10: Full energy consumption vs. time benchmarks against other YOLO-series [17, 18, 32] on the GPU (top) and NPU (bottom) for the nano (left), small (middle), and medium (right) model scales on the PascalVOC dataset.

B Additional Energy Estimation Results on ModalAI Sentinel

To further evaluate the proposed two-stage energy estimator on a real robotic edge platform, we include additional results on the ModalAI Sentinel Development Drone in Figure 11. In this experiment, we use 100 sampled architectures from the proposed search space benchmarked on the CPU, GPU, and NPU of the ModalAI Sentinel drone. For fairness, both the joint model and the two-stage model use the same generic architecture/device encoder. The architecture encoder one-hot encodes the XiResidualOFA design choices, while the device encoder one-hot encodes the target hardware. Energy values are normalized with respect to the range of each device. The joint model receives the concatenated architecture and device encoding and is pretrained on two devices before being fine-tuned on the target device. The proposed two-stage model first pretrains a generic predictor using only architecture encoding on the other two devices, then freezes this base predictor and fine-tunes a device-specific residual model on the target device using both architecture and device encoding. We evaluate few-shot adaptation from 2 to 20 fine-tuning samples, repeat each setting over 30 runs, and use the remaining target-device data as the test set [14, 39].

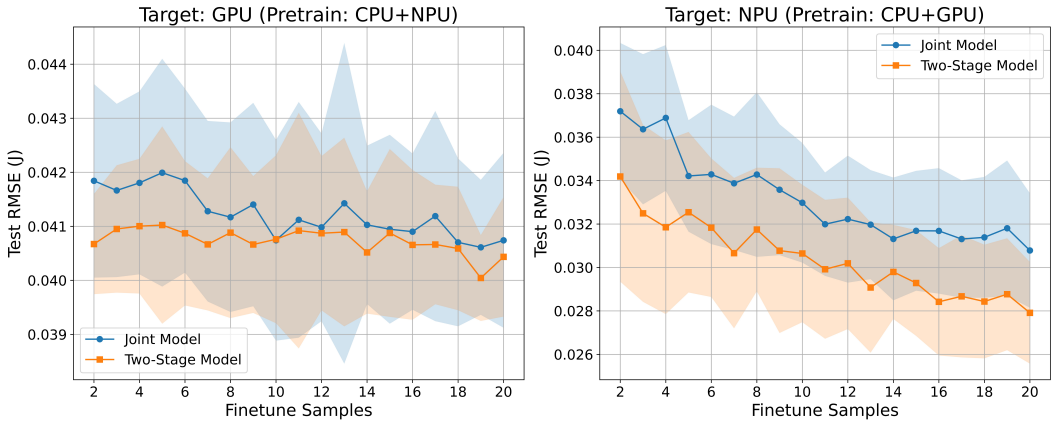


Figure 11: Two-Stage Energy Estimator vs. Joint Model on few data points from 2–20 samples on the ModalAI Sentinel Development Drone benchmark on CPU, GPU, and NPU using the proposed search space.

The results show that the two-stage model matches the joint model on CPU and outperforms it on GPU and NPU, with the clearest gains in the low-data regime. This supplementary experiment provides additional evidence that the proposed formulation remains effective on a real heterogeneous robotic platform, where accurate hardware energy measurements are especially limited.

C Architecture Ablation

We perform an ablation study to isolate the effect of the two main design changes in the proposed block: replacing the default YOLOv12 bottleneck with XiNet-style convolution and adding a residual connection. Specifically, we compare three detector variants: the original YOLOv12n bottleneck design (**Base**), a variant with only XiNet convolution (**XiConv**), and the proposed residual XiNet-based block (**XiRes**) [32, 2, 16]. Table 1 reports results on PascalVOC using small models deployed on GPU.

The ablation shows that introducing XiNet convolution yields the largest energy reduction, decreasing energy from **21.3 J** to **11.3 J** per 30 inferences, with a small drop in detection performance. Adding the residual connection recovers and slightly improves accuracy, yielding the best **84.38 mAP50** and **65.79 mAP50-95** while preserving the low-energy behavior at **11.1 J**. These results indicate that XiNet convolution is primarily responsible for the efficiency gain, while the residual connection improves optimization and restores accuracy, together motivating the final XiRes design used in the search space.

Table 1: Ablation of the proposed block design. Base is the default YOLOv12 [32] bottleneck, XiConv adds XiNet convolution [2], and XiRes further adds a residual connection [16]. Results are measured on small models running on GPU on the PascalVOC dataset [11].

	Base	**XiConv**	**XiRes**
XiNet Convolution		✓	✓
Residual Connection			✓
mAP50 (%)	84.34	84.26	**84.38**
mAP50-95 (%)	65.69	65.15	**65.79**
Energy / 30 Inf (J)	21.3± 0.93	11.3± 1.1	**11.1± 0.8**

D Additional Results on Compound Scaling and Average Power

To further evaluate the effect of compound scaling, we compare the average power of scaled XiYOLO models against YOLO baselines at the small and medium scales. The goal of this experiment is to

test whether scaling the searched base architecture can preserve strong detection accuracy while also reducing deployment power across different hardware platforms. Since higher-accuracy detectors generally require more computation, accuracy and energy usage are often positively correlated. This makes compound scaling a useful test of whether the searched architecture can maintain a favorable tradeoff as model capacity increases.

Table 2 reports results on the COCO dataset for the Qualcomm Adreno 650 GPU, the 15 TOPS NPU, and the NVIDIA Jetson Orin Nano. At the small scale, XiYOLOs reduces average power relative to both YOLOv8s and YOLO12s on all three platforms, while remaining competitive in accuracy. At the medium scale, XiYOLOm again achieves the lowest average power across all three devices, while preserving accuracy close to the stronger YOLO baselines. These results show that compound scaling of the searched base architecture can effectively limit the loss in detection accuracy while still reducing deployment power, supporting the use of XiYOLO as a scalable detector family for heterogeneous edge devices.

Table 2: Average power comparison for scaled detector variants on COCO. Results are reported on the Qualcomm Adreno 650 GPU, 15 TOPS NPU, and NVIDIA Jetson Orin Nano. XiYOLO achieves the lowest average power at both the small and medium scales while maintaining competitive accuracy. **Bold** indicates best while underline indicates second best.

Method	**mAP50-95**	**Average Power during Inference**		
		Adreno 650 (W)	**15 TOPS NPU (W)**	**Orin Nano (W)**
YOLOv8s	44.9	0.82	1.1	12.6
YOLO12s	**48.0**	1.4	1.5	12.4
XiYOLOs	46.5	**0.73**	**0.78**	**10.9**
YOLOv8m	50.2	2.5	2.5	13.5
YOLO12m	**52.5**	2.0	2.6	13.8
XiYOLOm	50.7	**1.5**	**1.6**	**12.6**

E Latency–Energy Tradeoff

Figure 12 compares XiYOLO and YOLO12 in the latency–energy plane on the Qualcomm Adreno 650 GPU, the 15 TOPS NPU, and the NVIDIA Jetson Orin Nano. Across all three devices, XiYOLO consistently achieves lower energy at similar latency. The gap is especially clear on the Adreno 650 GPU and 15 TOPS NPU, where XiYOLO remains well below YOLO12 across all operating points, while on the Jetson Orin Nano the improvement is smaller but still consistent. These results show that XiYOLO provides a better deployment tradeoff by reducing energy without introducing a meaningful latency penalty.

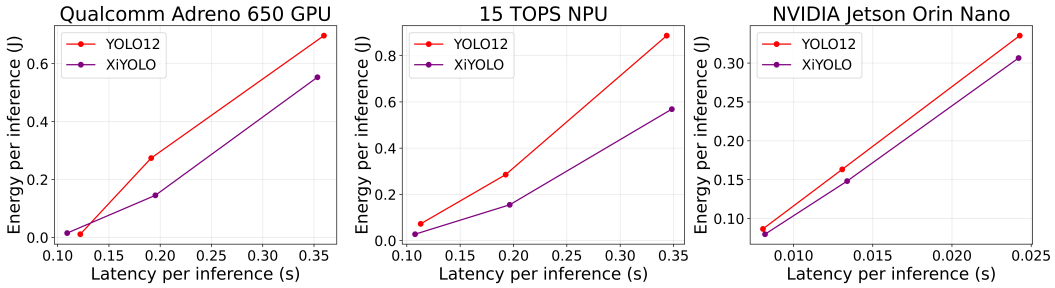


Figure 12: Latency–energy comparison between XiYOLO and YOLO12 on the Qualcomm Adreno 650 GPU, 15 TOPS NPU, and NVIDIA Jetson Orin Nano. XiYOLO consistently uses less energy at similar latency across all three platforms.