# XiYOLO: Energy-Aware Architecture Search for Object Detection from Sparse Hardware Measurements

**Anonymous authors**
Paper under double-blind review

## 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 multi-objective architecture search. The search yields an accuracy–energy Pareto frontier from which an operating point can be selected according to mission-specific energy requirements. We select a balanced operating point from this frontier as the reference architecture and apply post-search scaling to construct the XiYOLO family. 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. With only 20 target-device measurements, the proposed two-stage estimator recovers **88.6%** and **93.0%** of dense-reference search hypervolume on COCO and PascalVOC, respectively, corresponding to **38.1%** and **27.6%** improvements over joint adaptation.

## 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 Lin et al. (2020); Burrello et al. (2021); Banbury et al. (2021); Moss et al. (2023). 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 Tan et al. (2020); Xiong et al. (2021); Tu et al. (2023); Alqahtani et al. (2025). 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 Cai et al. (2018); Tan et al. (2019); Chen et al. (2019); Ghiasi et al. (2019); Wang et al. (2020); Sakuma et al. (2023), but applying NAS to energy-adaptive 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 Xiong et al. (2021); Gupta et al. (2024); Chen et al. (2025); Tran et al. (2026). 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 Cai et al. (2017); Guo et al. (2025); Zhang et al. (2025). The central challenge is therefore not only how to search for efficient detectors, but how to make energy-aware



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

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, efficiently searching that space, and enabling energy-aware search when hardware energy data is scarce. We address these challenges with a two-stage energy estimator for sparse target-device calibration, together with the XiResOFA search space and an iterative search strategy over the backbone, FPN, and PAN. Together, these components produce an accuracy–energy Pareto frontier spanning multiple deployment operating points. Mission requirements can then guide the selection of an efficiency- or accuracy-oriented architecture, while a balanced operating point is used as the reference design for post-search scaling into the XiYOLO family.

Our contributions are summarized as follows:

- We develop a **two-stage energy estimator for sparse target-device supervision**, combining a transferable architecture-level energy predictor with lightweight hardware-specific residual adaptation. With only 20 Adreno 650 measurements, the estimator recovers **88.6%** and **93.0%** of dense-reference search hypervolume on COCO and PascalVOC, respectively, compared with **64.1%** and **72.9%** for joint adaptation.
- We formulate detector search as a multi-objective accuracy–energy problem within stage-wise iterative search, maintaining a Pareto set of architectures spanning different deployment operating points.
- We introduce the XiResOFA energy-aware detector search space and scale a selected Pareto reference architecture into nano, small, and medium XiYOLO family. On PascalVOC, XiYOLOm 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.

## 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 Liu et al. (2016); Redmon et al. (2016); Lin et al. (2017); Tian et al. (2022); Tan et al. (2020). For resource-constrained deployment, lightweight detectors such as MobileDets, TinyissimoYOLO, and FemtoDet trade detection quality against computation, memory, or energy cost Xiong et al. (2021); Moosmann et al. (2023); Tu et al. (2023). In aerial and onboard settings, recent efforts have further emphasized efficient detection under limited compute and energy budgets Zhou et al. (2023); Liew et al. (2025); Nghiem et al. (2025); Geng et al. (2025). 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 deployment-dependent costs such as latency or memory into the search objective Cai et al. (2018); Tan et al. (2019); Wu et al. (2019). In object detection, prior work has searched detector backbones, feature pyramids, and full detector pipelines, including DetNAS, NAS-FPN, NAS-FCOS, MobileDets, and DetOFA Chen et al. (2019); Ghiasi et al. (2019); Wang et al. (2020); Xiong et al. (2021); Sakuma et al. (2023). More recent work has moved toward edge- and hardware-aware detector NAS Gupta et al. (2024); Chen et al. (2025); Tran et al. (2026). We adopt the stage-wise iterative search mechanism of ELASTIC Tran et al. (2026) and extend its candidate selection from accuracy-oriented search to a multi-objective accuracy–energy formulation that maintains a non-dominated set of deployment operating points.

**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 Cai et al. (2017); Guo et al. (2025); Zhang

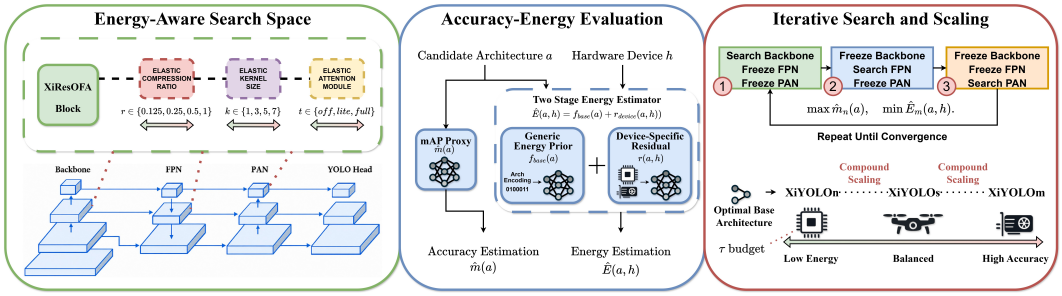



Figure 1: **XiYOLO energy-aware architecture search.** Candidate detectors from the XiResOFA search space are evaluated using an mAP proxy and a hardware energy proxy. Multi-objective search identifies an accuracy–energy Pareto frontier, from which a reference architecture is selected and scaled into the XiYOLO family.

et al. (2025). These studies show 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 Cai et al. (2017); Guo et al. (2025). This limitation is important for detector NAS, where dense hardware measurements across a large search space are costly to obtain. We therefore separate transferable architecture-level energy structure from target-device residual adaptation, enabling adaptation from 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 Lin et al. (2020); Burrello et al. (2021); Banbury et al. (2021); Moss et al. (2023). 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 detector architectures that jointly maximize detection quality and minimize deployment energy. In practice, accuracy is evaluated from the nano-scale realization, $\hat{m}_n(a)$, while energy is evaluated from the corresponding medium-scale realization, $\hat{E}_m(a,h)$. We therefore formulate architecture search as jointly maximizing $\hat{m}_n(a)$ and minimizing $\hat{E}_m(a,h)$ over $a \in \mathcal{A}$.

The search identifies the accuracy–energy Pareto frontier, from which a deployment operating point can be selected according to mission requirements. Given a deployment budget $\tau_{\text{mission}}$, the highest-accuracy Pareto architecture satisfying $\hat{E}_m(a,h) \leq \tau_{\text{mission}}$ is selected. When no explicit budget is specified, we select the balanced reference architecture and derive the XiYOLO family by post-search scaling. As shown in Figure 1, candidate detectors are sampled from the energy-aware search space over the backbone, FPN, and PAN, then evaluated using an mAP proxy and a hardware energy proxy. These signals guide the iterative NAS procedure toward the Pareto frontier, after which the selected reference architecture is scaled to different deployment regimes.

## 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 He et al. (2016), but it does not directly expose elastic operator-level choices suitable for energy-aware detector search. XiResOFA preserves a residual shortcut while introducing configurable architectural choices compatible with once-for-all search Cai et al. (2020). This design makes the block both trainable and searchable, while allowing its internal structure to adapt to different accuracy–energy operating points.

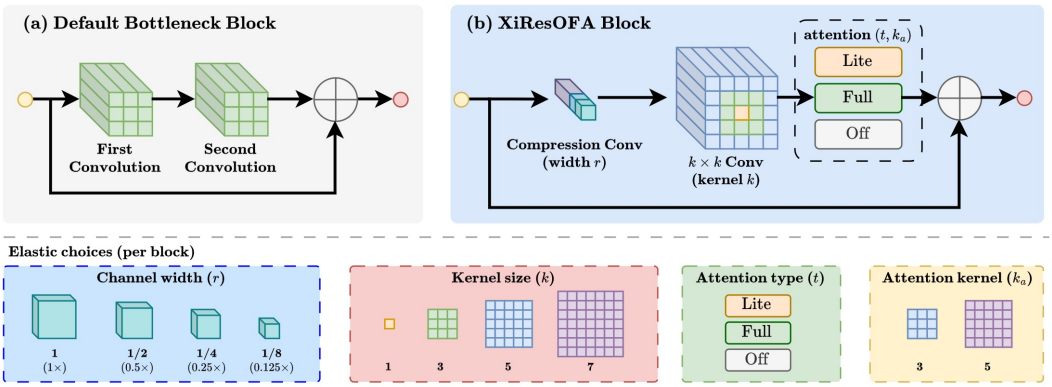



Figure 2: **Standard bottleneck and XiResOFA block with elastic architectural choices.** (a) the standard bottleneck block He et al. (2016) and (b) the proposed XiResOFA block, which preserves the residual connection and introduces searchable channel width $r$, kernel size $k$, attention type $t$, and attention kernel size $k_a$.

Figure 2 illustrates the four elastic choices exposed by each XiResOFA block. Each block is parameterized by a tuple $b = (r, k, t, k_a)$, where $r \in 1, 0.5, 0.25, 0.125$ is the retained channel-width ratio, $k \in 1, 3, 5, 7$ is the kernel size, and $t \in$ off, lite, full is the attention type, and $k_a \in 3, 5$ is the attention kernel size when attention is enabled. These dimensions induce clear accuracy–energy tradeoffs. Larger channel-width ratios preserve more channels and typically improve representational strength, but increase 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 applies the $k_a \times k_a$ attention convolution at the full channel width $C$, whereas lite attention introduces a bottleneck such that the convolution operates on $C/r_a$ channels, with $r_a \in 1, 2, 4, 8$, reducing the attention-branch cost by up to $8\times$. By embedding these choices within a shared supernet, the search space can span a range of candidates from aggressively efficient to higher-capacity detectors Cai et al. (2020); Tan et al. (2019); Wu et al. (2019).

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 Lin et al. (2017); Liu et al. (2018). 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 $\mathcal{A} = \mathcal{A}_{\text{back}} \times \mathcal{A}_{\text{fpn}} \times \mathcal{A}_{\text{pan}}$. 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 Cai et al. (2020); Sakuma et al. (2023).

### 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 Cai et al. (2017); Guo et al. (2025); Zhang et al. (2025). 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 Cai et al. (2017); Guo et al. (2025). For object detectors, this makes direct energy optimization particularly difficult, since exhaustive profiling over a large search space is costly and often impractical.

Figure 3 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 decompose energy prediction into a generic

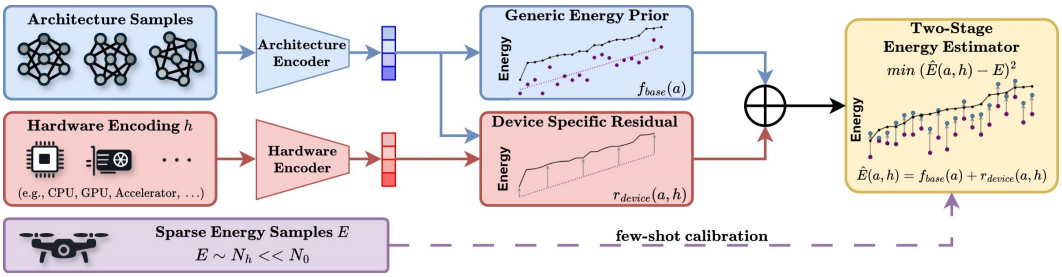



Figure 3: Overview of the proposed two-stage energy estimator. Sparse energy samples from the target hardware are combined with a device-agnostic architecture prior to learn a hardware-specific residual correction.

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 architecture-level energy prior and $r(a,h)$ is a lightweight hardware-specific residual model. Source-device energies are independently normalized to $[0,1]$ before pretraining, allowing $f_{\text{base}}$ to learn a device-agnostic architecture ranking. 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 Guo et al. (2025); Zhang et al. (2025). 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 sparse target-device 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 Chen et al. (2019); Ghiasi et al. (2019); Wang et al. (2020); Sakuma et al. (2023). 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 Lin et al. (2017); Liu et al. (2018). To exploit this structure, we build on the iterative search strategy of Tran et al. (2026), progressively refining detector components rather than performing a single monolithic joint search, and extend its candidate ranking to the multi-objective accuracy–energy setting.

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, using the two objectives $\mathbf{f}(a) = (-\hat{m}(a), \hat{E}(a,h))$. Candidates are ranked by Pareto dominance and crowding distance, and mutation, crossover, and a population-passthrough ratio of $0.5$ balance exploration with retention across backbone, FPN, and PAN updates. A global nondominated archive preserves strong accuracy–energy solutions across phases, while the search proceeds under a fixed evaluation budget. This procedure reduces search complexity while preserving the main accuracy–energy tradeoff.

The search produces an accuracy–energy Pareto frontier. To obtain a balanced reference design, we min–max normalize both objectives and select the Pareto point with the maximum perpendicular distance from the line joining the minimum-energy and maximum-accuracy extremes. This point is used as the default reference architecture, while other Pareto points can be selected for different deployment requirements.

After search, we derive a detector family by scaling the selected reference architecture to different deployment budgets. Let $S(a^\star, \alpha)$ denote a scaling operator with scale factor $\alpha$. We obtain the final variants as $a^{(\ell)} = S(a^\star, \alpha_\ell)$ for $\ell \in \{n, s, m\}$, where $n$, $s$, and $m$ denote nano, small, and medium variants. Post-search scaling adjusts width and maximum channel capacity to produce higher- and lower-budget variants Cai et al. (2020). This way, iterative NAS identifies the accuracy–energy tradeoff, and post-search scaling adapts the selected reference design to different operating regimes without requiring separate searches for each deployment point.

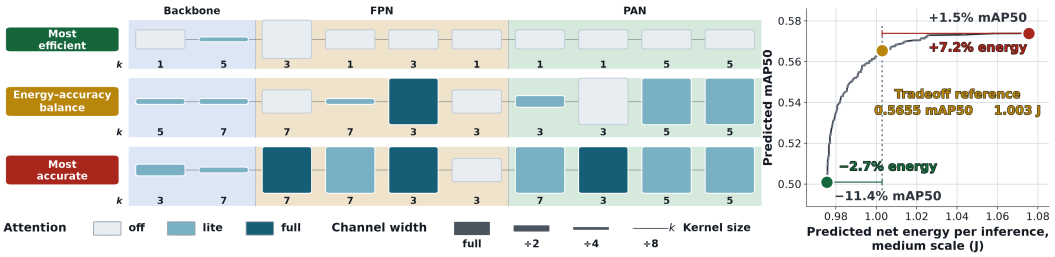



Figure 4: COCO Pareto search results. Left: representative architectures at the most energy-efficient, balanced, and most accurate operating points. Right: predicted Pareto frontier using nano-scale mAP50 and medium-scale predicted energy per inference. The balanced point is selected as the reference architecture for scaling.

# 4 Experiments

## 4.1 Experimental Setup

**Training protocol.** We construct XiYOLO from a YOLOv12n-based supernet using the XiResOFA search space defined in Section 3.2. On PascalVOC Everingham et al. (2010), we train the supernet for 300 epochs on one RTX PRO 6000 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 sample nano-scale subnetworks to train the mAP proxy, while the reference energy proxy is trained separately from approximately 900 measured medium-scale XiResOFA architectures, with a fixed 100-architecture holdout for evaluation. Both estimators are implemented as MLPs with hidden size 400. Supernet training requires approximately **19.7 GPU-h** on PascalVOC and **204.5 GPU-h** on COCO.

**Search configuration.** We run five independent Pareto searches with a budget of 20,000 unique architecture evaluations per run and a population size of 50. The reported frontier is obtained from the nondominated union of the five runs. The energy objective for the primary architecture search uses the densely calibrated Adreno 650 reference energy proxy. This high-fidelity proxy isolates the attainable performance of the search space and search procedure from errors due to sparse energy calibration. Accuracy is evaluated from the nano-scale supernet, while energy is predicted for the corresponding medium-scale realization because energy differences among candidate architectures at nano scale are close to measurement noise and yield unreliable rankings. Section 4.5 separately evaluates sparse target-device calibration using only $N_h = 20$ Adreno measurements. Additional details are provided in Appendix B.

**Deployment platforms and baselines.** We evaluate XiYOLO against YOLOv8 and YOLOv12 Tian et al. (2025) on the ModalAI Sentinel Development Drone, with a Qualcomm QRB5165 CPU, Adreno 650 GPU, and 15 TOPS NPU. Models are exported to FP16 TFLite. System power is measured at the drone input using a $10\,\mathrm{m}\Omega$ shunt resistor and a Rigol DHO804 oscilloscope. Baseline system power is subtracted from the network-on measurement, and net energy per inference is averaged over 1000 inferences.

## 4.2 Pareto Search and Reference Architecture

We first examine the architectures produced by the Pareto search before full training and scaling. Figure 4 shows the predicted accuracy–energy Pareto frontier together with three representative operating points: the most energy-efficient architecture, the balanced reference architecture, and the most accurate architecture. The mAP values are predicted from the nano-scale realization, while the energy values are predicted from the corresponding medium-scale realization.

The balanced reference architecture achieves a predicted **0.5655 mAP50** at **1.003 J** per inference. Relative to this point, the most accurate architecture improves predicted mAP50 by only **1.5%** while increasing predicted energy by **7.2%**, whereas the most energy-efficient architecture reduces predicted energy by **2.7%** with an **11.4%** reduction in predicted mAP50. We therefore select the

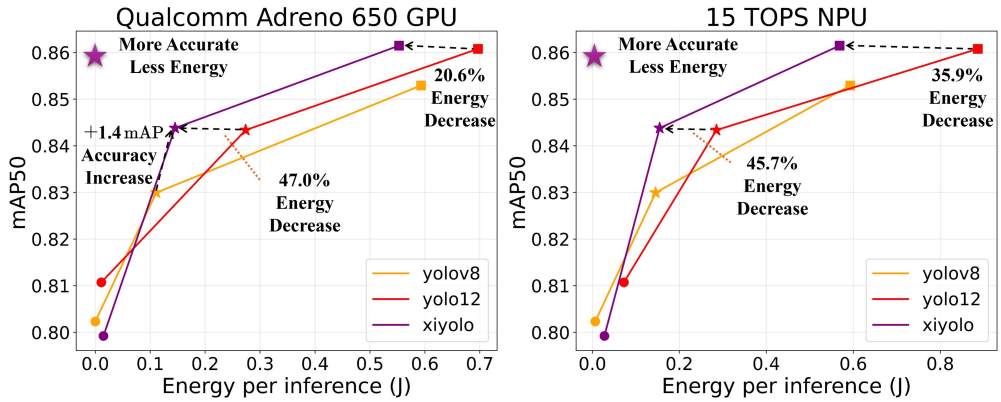



Figure 5: Accuracy–energy operating points of XiYOLO and YOLO baselines Jocher (2026); Tian et al. (2025) on PascalVOC using the GPU (left) and NPU (right).

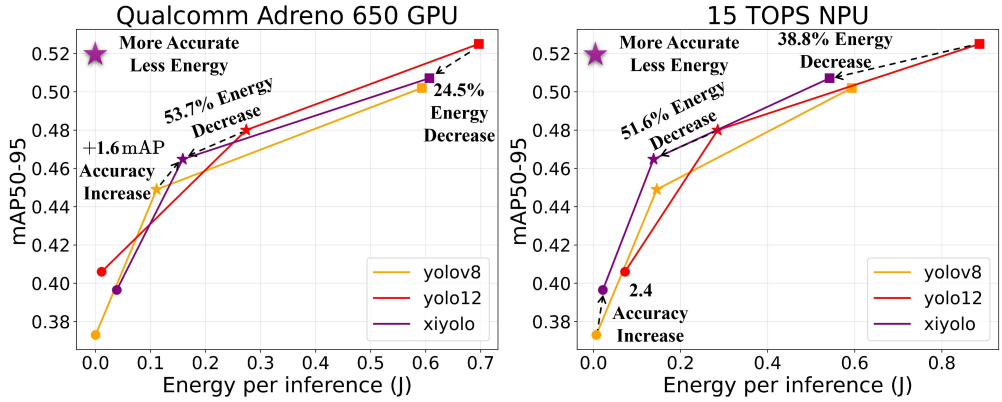



Figure 6: Accuracy–energy operating points of XiYOLO and YOLO baselines Jocher (2026); Tian et al. (2025) on COCO using the GPU (left) and NPU (right).

balanced reference architecture for post-search scaling, while the remaining Pareto points represent alternative candidate deployment operating points.

### 4.3 Accuracy and Energy Tradeoffs

We next evaluate the fully trained and scaled XiYOLO models. 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 clearest accuracy–energy gains appear at the small and medium scales.

On PascalVOC, XiYOLO shows favorable tradeoffs against YOLOv8 and YOLOv12 on both GPU and NPU. At the medium scale, XiYOLO reaches **86.15 mAP50**, effectively matching YOLOv12m accuracy while reducing energy by **20.6%** on GPU and **35.9%** on NPU. At the small scale, XiYOLO reaches **84.38 mAP50**, matching YOLOv12s at **84.34 mAP50** while reducing energy by **47.0%** on GPU and **45.7%** on NPU. Relative to YOLOv8, XiYOLO improves mAP50 by **+1.4 points** at the small scale. These results show the clearest gains at the small and medium operating points.

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 reduces energy relative to YOLOv12s by **53.7%** on GPU and **51.6%** on NPU, while improving accuracy over YOLOv8s by **+1.6 mAP50-95 points**. At the nano scale, XiYOLO

Table 1: **Surrogate validation.** Accuracy uses mAP50; energy uses mJ. Target device: Adreno 650.

| Proxy | Dataset | RMSE ↓ | Spearman $\rho$ ↑ | Kendall $\tau$ ↑ | Recall@10% ↑ |
|---|---|---|---|---|---|
| Accuracy | COCO | 0.0197 | 0.7503 | 0.5607 | 0.400 |
| Accuracy | PascalVOC | 0.0099 | 0.8853 | 0.7127 | 0.600 |
| Energy | Adreno 650 | 25.25 | 0.902 | 0.733 | 0.500 |

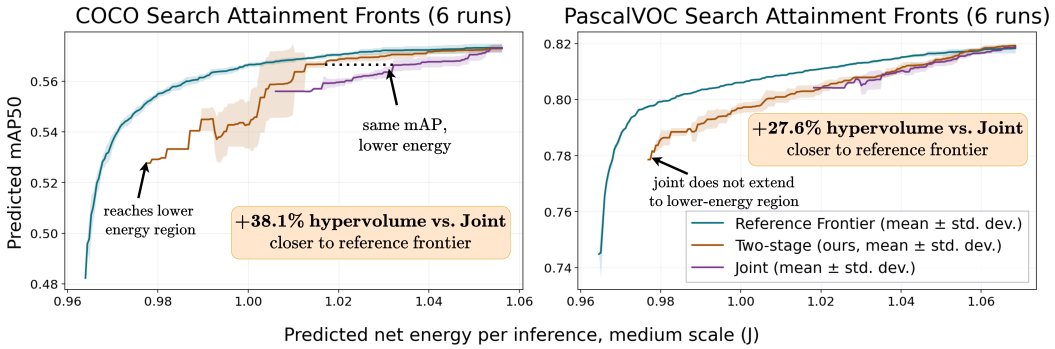



Figure 7: **Sparse-calibrated architecture search.** Search attainment fronts on COCO (left) and PascalVOC (right) using the reference, two-stage ($N_h = 20$), and joint ($N_h = 20$) energy proxies. Curves show mean ± standard deviation over six matched random seeds.

further improves accuracy over YOLOv8n by **+2.4 mAP50-95 points**. Overall, Figures 5 and 6 show favorable measured operating points, with the clearest energy reductions against YOLOv12 at the small and medium scales. Full cumulative-energy results, cross-device scaling behavior, and latency–energy comparisons are provided in Appendices E, G, and H, respectively.

### 4.4 Search-Surrogate Validation

We next validate the accuracy and energy surrogates on held-out architectures excluded from predictor fitting, directly assessing their reliability for candidate ranking during multi-objective search.

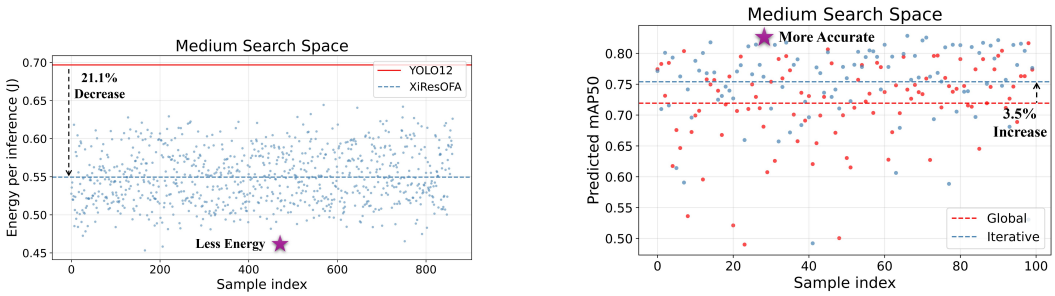
**Accuracy proxy.** For each dataset, 1000 nano-scale subnetworks are evaluated using inherited supernet weights and split into 720 training, 80 validation, and 200 held-out test architectures. As shown in Table 1, the resulting proxy preserves architecture ordering, with Spearman correlations of **0.750** on COCO and **0.885** on PascalVOC and top-10% recall of **40%** and **60%**, respectively.

**Energy proxy.** We evaluate the medium-scale Adreno 650 reference energy proxy on a fixed 100-architecture holdout excluded from fitting. Table 1 shows that, when trained from approximately 900 measured architectures, the predictor reaches a Spearman correlation of **0.902** and Kendall's $\tau$ of **0.733**, providing a strong reference ranking signal for the reported architecture search.

### 4.5 Sparse Energy Calibration for Architecture Search

We next evaluate whether sparse target-device calibration preserves useful energy-aware search behavior. We compare three Adreno 650 energy proxies: a densely calibrated reference trained from approximately 900 target-device measurements, the proposed two-stage estimator adapted using only $N_h = 20$ measurements, and a joint estimator adapted using the same 20 measurements. All methods use the same accuracy proxy and iterative search configuration, with 20,000 architecture evaluations over six matched random seeds. For a common comparison, all returned architectures are re-scored using the reference energy proxy.

Figure 7 shows that the two-stage estimator produces search fronts substantially closer to the dense reference than joint adaptation, particularly in the lower-energy regime. Under the same 20-measurement budget, Two-stage recovers 88.6% and 93.0% of dense-reference search hypervolume on COCO and PascalVOC, compared with 64.1% and 72.9% for joint corresponding to improve-




(a) XiResOFA energy distribution.

(b) Global vs. iterative search accuracy.

Figure 8: **Search-space characteristics.** (a) Energy of sampled XiResOFA models. (b) Predicted accuracy under global and iterative search.

ments of $38.1\%$ and $27.6\%$, respectively. The corresponding sparse energy ranking is also stronger, with Spearman correlation improving from 0.693 to 0.761. These results show that the transferable energy prior translates sparse target-device calibration into better architecture selection during NAS. Full proxy and search metrics are provided in Appendix C.

### 4.6 Search Space Characteristics

We next examine the properties of the proposed medium-scale search space in terms of both energy and accuracy. Figure 8(a) shows the energy distribution of randomly sampled XiResOFA models. Compared with the YOLOv12 medium baseline, the sampled models occupy a lower-energy region, with average energy reduced by **21.1%**. This indicates that the proposed search space is biased toward more energy-efficient designs while spanning a broad set of candidate architectures.

Figure 8(b) compares the predicted accuracy of architectures sampled by the global and iterative search procedures. Architectures sampled by the iterative procedure have about **3.5%** higher average predicted mAP50 than those sampled by global search. This distributional comparison shows that the iterative procedure systematically concentrates evaluations in a higher-performing proxy-predicted region. Consistent with this trend, the single-seed comparison in Appendix D selects an iterative-search architecture with $0.4617$ mAP50-95 and $154.1$ mJ per inference compared with $0.4473$ and $163.6$ mJ for global search under the same $20,000$-evaluation budget. Appendix F further isolates the contributions of the XiRes block components.

## 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 multi-objective 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. The search produces an accuracy–energy Pareto frontier, from which a balanced reference architecture is selected and scaled into the XiYOLO family. On PascalVOC, XiYOLO improves over YOLOv12 at the small and medium scales while substantially reducing energy on GPU and NPU. On COCO, XiYOLO again delivers large energy reductions relative to YOLOv12, especially at the small scale, while maintaining competitive medium-scale accuracy. Using 20 target-device measurements, the proposed two-stage estimator recovers **88.6%** and **93.0%** of the calibrated reference-search hypervolume on COCO and PascalVOC, respectively, demonstrating that sparse hardware calibration can preserve most of the energy-aware search quality.

Our study also has several limitations: the search is confined to a YOLOv12-derived detector family, the reported XiYOLO family is obtained by scaling a selected Pareto reference architecture rather than independently optimizing each model scale, the search jointly uses nano-scale accuracy and medium-scale energy signals, and the deployment evaluation focuses on selected GPU and NPU settings. Despite these limitations, our results show that energy-aware architecture search can produce detector families with favorable accuracy–energy tradeoffs for heterogeneous edge deployment.

## REFERENCES


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 (eds.), *Service-Oriented Computing*, pp. 142–150, Singapore, 2025. Springer Nature. ISBN 978-981-96-0805-8. doi: 10.1007/978-981-96-0805-8_11.

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

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.

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.

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*, pp. 622–637. PMLR, November 2017. URL https://proceedings.mlr.press/v77/cai17a.html.

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.

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.

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/.

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.

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.

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.

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)*, pp. 7029–7038, June 2019. doi: 10.1109/CVPR.2019.00720. URL https://ieeexplore.ieee.org/document/8954436. ISSN: 2575-7075.

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.

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)*, pp. 1005–1010, June 2024. doi: 10.1109/CAI59869.2024.00183. URL https://ieeexplore.ieee.org/document/10605392.

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)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90. URL https://ieeexplore.ieee.org/document/7780459. ISSN: 1063-6919.

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

Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, January 2023. URL https://github.com/ultralytics/ultralytics. original-date: 2022-09-11T16:39:45Z.

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.

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, pp. 11711–11722. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html.

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)*, pp. 936–944, July 2017. doi: 10.1109/CVPR.2017.106. URL https://ieeexplore.ieee.org/document/8099589. ISSN: 1063-6919.

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*, pp. 8759–8768, June 2018. doi: 10.1109/CVPR.2018.00913. URL https://ieeexplore.ieee.org/document/8579011. ISSN: 2575-7075.

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 (eds.), *Computer Vision – ECCV 2016*, pp. 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0. doi: 10.1007/978-3-319-46448-0_2.

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)*, pp. 1–5, June 2023. doi: 10.1109/AICAS57966.2023.10168657. URL https://ieeexplore.ieee.org/document/10168657. ISSN: 2834-9857.

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, pp. 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.

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.

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)*, pp. 779–788, June 2016. doi: 10.1109/CVPR.2016.91. URL https://ieeexplore.ieee.org/document/7780460. ISSN: 1063-6919.

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)*, pp. 1325–1334, October 2023. doi: 10.1109/ICCVW60793.2023.00143. URL https://ieeexplore.ieee.org/document/10350829. ISSN: 2473-9944.

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)*, pp. 2820–2828, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Tan_MnasNet_Platform-Aware_Neural_Architecture_Search_for_Mobile_CVPR_2019_paper.

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)*, pp. 10778–10787, June 2020. doi: 10.1109/CVPR42600.2020.01079. URL https://ieeexplore.ieee.org/document/9156454. ISSN: 2575-7075.

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.

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.

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.

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)*, pp. 13272–13281, October 2023. doi: 10.1109/ICCV51070.2023.01225. URL https://ieeexplore.ieee.org/document/10376762. ISSN: 2380-7504.

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)*, pp. 11940–11948, June 2020. doi: 10.1109/CVPR42600.2020.01196. URL https://ieeexplore.ieee.org/document/9156326. ISSN: 2575-7075.

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)*, pp. 10726–10734, June 2019. doi: 10.1109/CVPR.2019.01099. URL https://ieeexplore.ieee.org/document/8953587. ISSN: 2575-7075.

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)*, pp. 3824–3833, June 2021. doi: 10.1109/CVPR46437.2021.00382. URL https://ieeexplore.ieee.org/document/9578555. ISSN: 2575-7075.

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.

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 AI USE DISCLOSURE

Generative AI such as ChatGPT and Claude were used to assist with manuscript editing, including improving wording and readability, reorganizing portions of the presentation, and providing feedback on the clarity and structure of the paper. They were also used for brainstorming and literature-search assistance, as well as for coding assistance during implementation. All AI-assisted code was reviewed and validated by the authors before being used in the reported experiments. All technical claims, citations, experimental results, and AI-assisted text were likewise reviewed and verified by the authors. Generative AI tools were not used to generate experimental data or formulate the core methodology. The authors take full responsibility for the final content, implementation, and results of this work.

# B SEARCH AND REPRODUCIBILITY DETAILS

Each primary search uses a population size of 50 and a budget of 20,000 unique architecture evaluations. We perform five independent runs for each dataset and report the nondominated union of their final archives. Candidate architectures are ranked using Pareto dominance and crowding distance under the predicted accuracy and energy objectives. Within each active stage, parents are selected by binary tournament on Pareto rank and crowding distance, with uniform crossover probability $0.9$, coordinate-wise mutation probability $p_{\mathrm{mut}} = 0.15$, and NSGA-II survival over the combined parent–offspring population. Variation is restricted to the active stage. The backbone (blocks 1–2), FPN (blocks 3–6), and PAN (blocks 7–10) are searched cyclically for six complete cycles. At each stage transition, the population-passthrough ratio of $0.5$ retains 25 architectures from the current nondominated archive and introduces 25 uniformly sampled architectures. Only unique architectures count toward the evaluation budget, and cached duplicates are not re-evaluated. Search terminates on budget exhaustion. The reported search uses the Qualcomm Adreno 650 GPU as the energy target with the densely calibrated reference energy proxy trained from approximately 900 target-device measurements.

For sparse-proxy evaluation, the generic architecture-level predictor is pretrained on Jetson AGX Orin, Jetson Orin Nano, and 15 TOPS NPU measurements, after which the hardware-specific residual is adapted using the same $N_h = 20$ Adreno calibration samples. The Joint model is pretrained on the same source data and fine-tuned end-to-end on the same 20 target examples. A fixed 100-architecture test set is excluded from all proxy fitting.

All three energy proxies are evaluated using the same accuracy proxy and 20,000-evaluation iterative search over six matched seeds $\{0, 42, 123, 7, 21, 99\}$. Returned architectures are re-scored using the reference energy proxy to provide a common energy axis for comparison.

The nano, small, and medium variants use depth multiplier, width multiplier, and maximum channel count $(d, w, c_{\max})$ equal to $(0.50, 0.25, 1024)$, $(0.50, 0.50, 1024)$, and $(0.50, 1.00, 512)$, respectively.

# C ENERGY-ESTIMATOR AND SPARSE-SEARCH RESULTS

We provide additional evaluation of the energy proxies used in the sparse target-device experiments. We first compare predictor quality on held-out Adreno 650 measurements, followed by the resulting architecture-search performance under the same sparse calibration budget.

## C.1 SPARSE ENERGY PROXY QUALITY

Table 2 compares the densely calibrated reference proxy with the proposed two-stage estimator and joint adaptation. The reference is trained from approximately 900 Adreno 650 measurements, while Two-stage and Joint each use only $N_h = 20$ target-device measurements. All methods are evaluated on the same fixed 100-architecture holdout excluded from fitting.

Under the same 20-measurement budget, the two-stage estimator provides stronger prediction and ranking quality than joint adaptation, including higher Spearman correlation ($0.761$ vs. $0.693$) and Kendall's $\tau$ ($0.576$ vs. $0.507$). Joint achieves lower single-architecture top-1 regret, defined as the

Table 2: **Adreno 650 energy-proxy validation.** Two-stage and Joint use the same $N_h = 20$ target-device measurements. Bold indicates the best metric values among these two sparse-calibration methods, including ties.

| Metric | Reference | Two-stage (Ours) | Joint |
|---|---|---|---|
| Target samples | 900 | 20 | 20 |
| MAE (mJ) ↓ | 20.13 | **26.04** | 34.09 |
| RMSE (mJ) ↓ | 25.25 | **33.09** | 43.84 |
| $R^2$ ↑ | 0.814 | **0.681** | 0.439 |
| Spearman $\rho$ ↑ | 0.902 | **0.761** | 0.693 |
| Kendall $\tau$ ↑ | 0.733 | **0.576** | 0.507 |
| Recall@10% ↑ | 0.500 | **0.500** | **0.500** |
| Regret (%) ↓ | 6.55 | 9.38 | **4.21** |

true-energy gap between the proxy's top-ranked architecture and the lowest-energy architecture, normalized by the energy range of the 100-architecture holdout. Because regret evaluates only the top-ranked architecture, it does not capture the broader ranking quality used during search.

### C.2 Sparse-Proxy Search Results

We next evaluate how these proxy differences affect architecture search. Reference, Two-stage, and Joint use the same accuracy proxy and iterative search configuration over six matched random seeds, and all returned architectures are re-scored using the reference energy proxy. Hypervolume is computed on the raw minimization objectives $(-m, E)$ using a shared reference point within each dataset. The reference point is defined as $r_{\mathrm{HV}} = (-m_{\min}, E_{\max})$, where $m_{\min}$ and $E_{\max}$ are the worst accuracy and energy values in the pooled comparison set formed from the non-dominated fronts of all three proxies across all six seeds. The same reference point is used for all three proxies within a dataset, so absolute hypervolume values are comparable within, but not across, datasets. We additionally report hypervolume relative to the dense-reference search.

Table 3: **Sparse-proxy search performance.** Hypervolume and minimum energy are reported as mean ± standard deviation over six matched searches. Relative HV is expressed as a percentage of dense-reference search hypervolume. Minimum energy is scored using the reference energy proxy. Bold indicates the better result between Two-stage and Joint.

| Energy proxy | Hypervolume ↑ | Relative HV (%) ↑ | Min. energy (J) ↓ |
|---|---|---|---|
| **COCO** | | | |
| Reference | $0.01638 \pm 0.00017$ | 100.0 | $0.9646 \pm 0.0006$ |
| Two-stage | $\mathbf{0.01451 \pm 0.00024}$ | **88.6** | $\mathbf{0.9909 \pm 0.0067}$ |
| Joint | $0.01051 \pm 0.00185$ | 64.1 | $1.0153 \pm 0.0049$ |
| **PascalVOC** | | | |
| Reference | $0.02731 \pm 0.00015$ | 100.0 | $0.9650 \pm 0.0002$ |
| Two-stage | $\mathbf{0.02539 \pm 0.00040}$ | **93.0** | $\mathbf{0.9816 \pm 0.0052}$ |
| Joint | $0.01990 \pm 0.00074$ | 72.9 | $1.0287 \pm 0.0071$ |

Two-stage consistently produces search outcomes closer to the densely calibrated reference than Joint. Relative to Joint, Two-stage improves hypervolume by **38.1%** on COCO and **27.6%** on PascalVOC, while also reaching lower-energy architectures under the common reference scoring. These results show that the improved sparse energy ranking translates directly into stronger architecture selection during NAS.

## D SUPPORTING SEARCH AND DEPLOYMENT COMPARISONS

Table 4 compares final architectures selected by iterative and global evolutionary search under a controlled comparison using the same supernet, predictors, accuracy and energy objectives, evolutionary settings, and 20,000-evaluation budget. This experiment uses one independent seed and is therefore reported as supporting rather than aggregate evidence.

Table 4: Final small-scale architectures selected by iterative and global search in one independent seed. Hardware values are mean ± standard deviation over 30 timed inferences on Jetson Orin Nano.

| Search | mAP50–95 | Latency (ms) | Energy/Inf. (mJ) | Total Power (W) |
|---|---|---|---|---|
| Iterative | **0.4617** | $12.73 \pm 0.02$ | **154.1±0.7** | **16.43±0.08** |
| Global | 0.4473 | $12.43 \pm 0.03$ | $163.6 \pm 0.7$ | $17.38 \pm 0.07$ |

Table 5 compares XiYOLOn with a 30% magnitude-based channel-pruned YOLOv12n. Pruning reduces latency but increases active energy in this deployment, whereas XiYOLOn reduces energy by 19.7% relative to the pruned model while improving mAP50–95 by 1.1 points.

Table 5: Magnitude-based channel-pruning comparison on Jetson Orin Nano.

| Model | Pruning | mAP50–95↑ | Latency (ms)↓ | Energy/Inf. (mJ)↓ | Total Power (W)↓ |
|---|---|---|---|---|---|
| YOLOv12n | 0% | 0.400 | $7.97 \pm 0.06$ | $81.0 \pm 0.7$ | $15.14 \pm 0.16$ |
| YOLOv12n | 30% | 0.385 | $7.42 \pm 0.02$ | $86.0 \pm 0.5$ | $16.04 \pm 0.08$ |
| XiYOLOn | 0% | 0.396 | $8.39 \pm 0.03$ | **69.1±0.6** | **13.32±0.15** |

Energy per inference satisfies $E = P_{\text{active}}t = (P_{\text{total}} - P_{\text{idle}})t$, where the tables report total system power and $P_{\text{idle}}$ denotes the measured idle baseline, so lower energy does not necessarily imply lower latency. Table 6 reports the corresponding Jetson measurements; the memory reductions are empirical outcomes rather than constraints imposed during search.

Table 6: Deployment metrics on Jetson Orin Nano.

| Model | Latency (ms)↓ | Peak memory (MB)↓ | Energy/Inf. (mJ)↓ | Total Power (W)↓ |
|---|---|---|---|---|
| YOLOv12n | $7.97 \pm 0.06$ | $181 \pm 34$ | $81.0 \pm 0.7$ | $15.14 \pm 0.16$ |
| XiYOLOn | $8.39 \pm 0.03$ | **157±16** | **69.1±0.6** | **13.32±0.15** |
| YOLOv12s | $13.25 \pm 0.09$ | $192 \pm 37$ | $162.2 \pm 1.7$ | $17.21 \pm 0.19$ |
| XiYOLOs | $13.58 \pm 0.09$ | **173±33** | **145.8±1.0** | **15.71±0.15** |

## E FULL DEPLOYMENT ENERGY RESULTS

We next examine how the searched architectures behave during sustained deployment. Figure 9 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 YOLOv8 and YOLOv12 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 YOLOv8 and YOLOv12 throughout the benchmark horizon, with **XiYOLO-VOC** achieving the lowest cumulative energy overall. By the end of the run, XiYOLO-COCO uses approximately **196 J** less cumulative energy than YOLOv8. On the NPU, the separation is even larger: both XiYOLO variants again remain below YOLOv8 and YOLOv12, with XiYOLO-COCO using approximately **232 J** less cumulative energy than YOLOv8. Overall, Figure 9 shows that the searched XiYOLO architectures achieve lower deployment-time energy consumption than the compared YOLOv8 and YOLOv12 medium-scale models on both accelerator backends.

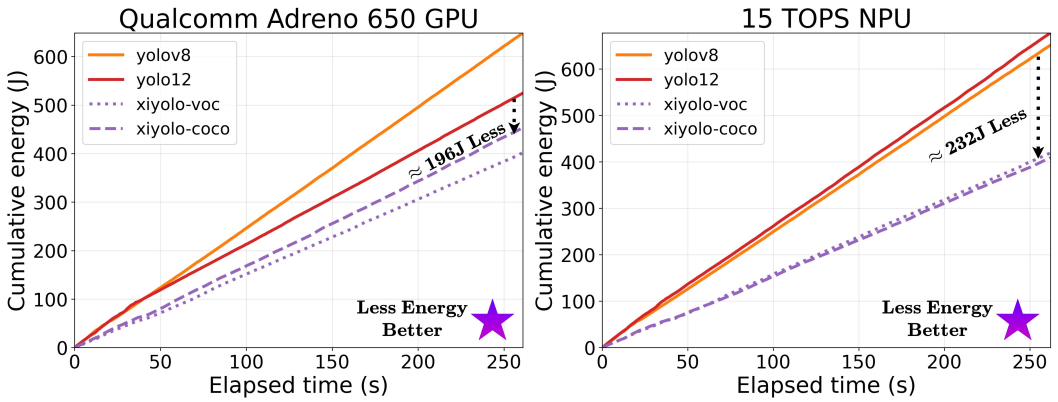



Figure 9: Energy consumption vs. time benchmarks against YOLOv8 and YOLOv12 Jocher et al. (2023); Tian et al. (2025) on the GPU (left) and NPU (right) at the medium scale.

## F 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**) Tian et al. (2025); Ancilotto et al. (2023); He et al. (2016). Table 7 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 **8.24 J** to **4.15 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 **4.36 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 7: Ablation of the proposed block design. Base is the default YOLOv12 Tian et al. (2025) bottleneck, XiConv adds XiNet convolution Ancilotto et al. (2023), and XiRes further adds a residual connection He et al. (2016). Results are measured on small models running on GPU on the PascalVOC dataset Everingham et al. (2010).

| | **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) | 8.24± 0.36 | 4.15± 0.41 | **4.36± 0.33** |

## G Additional Results on Post-Search Scaling and Average Power

To further evaluate the effect of post-search 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 post-search scaling a useful test of whether the searched architecture can maintain a favorable tradeoff as model capacity increases.

Table 8 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 YOLOv12s 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 post-search 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 8: 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 Active Power during Inference | | |
|---|---|---|---|---|
| | | Adreno 650 (W) | 15 TOPS NPU (W) | Orin Nano (W) |
| YOLOv8s | 44.9 | 0.82 | 1.1 | 12.6 |
| YOLOv12s | **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 |
| YOLOv12m | **52.5** | 2.0 | 2.6 | 13.8 |
| **XiYOLOm** | 50.7 | **1.5** | **1.6** | **12.6** |

## H Latency–Energy Tradeoff

Figure 10 compares XiYOLO and YOLOv12 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 YOLOv12 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.

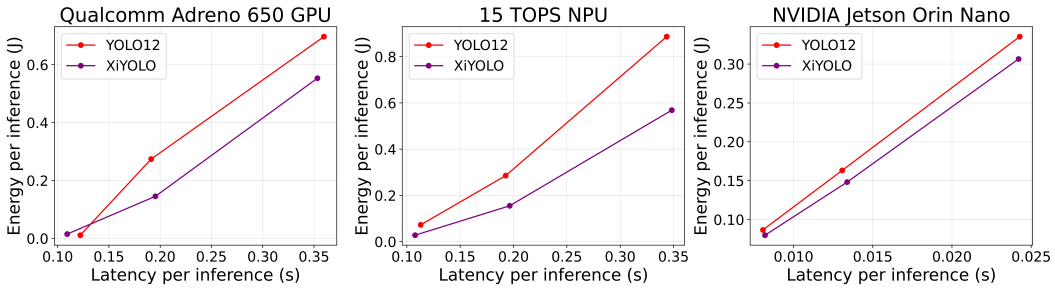



Figure 10: Latency–energy comparison between XiYOLO and YOLOv12 on the Qualcomm Adreno 650 GPU, 15 TOPS NPU, and NVIDIA Jetson Orin Nano. XiYOLO consistently uses less energy at similar latency across all three platforms.