EENAS: Zero-Shot Energy-Efficiency-Aware Neural Architecture Search

Zeyu Yang
Dept. of Engineering Science
University of Oxford
zeyu.yang@eng.ox.ac.uk

Wes Armour
Dept. of Engineering Science
University of Oxford
wes.armour@oerc.ox.ac.uk

Abstract

Despite the critical need for sustainable deep learning, model energy efficiency remains largely neglected in Hardware-Aware Neural Architecture Search (HW-NAS). Discovering energy-efficient models is severely bottlenecked by costly on-device measurements and the brittleness of training-free accuracy proxies, which, when used in isolation, risk favouring unbalanced "glass cannon" architectures. We propose Energy-Efficiency-Aware Neural Architecture Search (EENAS), a strictly zero-shot HW-NAS framework. EENAS introduces a PCA-based anomaly router that dispatches tasks to decision trees or tabular transformers, achieving <10% MAPE zero-shot energy prediction on entirely unseen GPUs. Simultaneously, we robustly estimate accuracy by ensembling training-free proxies via a novel Log Z-score aggregation. Combining these predictors unlocks Absolute Hardware Bounding, discovering Pareto-optimal architectures for strict energy budgets entirely offline in under 30 minutes. Evaluated on ImageNet-1K under a unified recipe, our models achieve 40% less energy consumption at comparable accuracy, or a 2.5% accuracy improvement under the same energy budget compared to efficient baselines.

1 Introduction

The rapid scaling of deep learning has placed AI on an increasingly unsustainable energy trajectory. Reports from major technology providers, including Google, Meta, and Microsoft, show electricity use growing at a geometric mean rate of roughly 20–30% year on year over the past five years, reaching tens of terawatt-hours annually by 2024 [8, 22, 23]. Beyond the direct costs of power and cooling, this trend increases carbon emissions and concentrates frontier AI development among a small number of organisations able to absorb these costs. Energy inefficiency also limits deployment in edge settings, where models must operate under strict battery and thermal design power constraints [39, 26].

Within this footprint, inference dominates. Between 2019 and 2021, Google reported that inference accounted for 60% of its AI energy usage [25], while Meta observed a 70% inference share [37]. With the ubiquitous deployment of generative AI, this disparity continues to widen. Optimizing absolute energy consumption directly during the architectural design phase is therefore critical.

Hardware-Aware Neural Architecture Search (HW-NAS) automates this co-design but traditionally suffers from extreme inefficiency. Early NAS methods required thousands of GPU hours [42], consuming megawatt-hours of electricity. While decoupled supernets like Once-for-All [3] reduced search costs, adapting HW-NAS to new hardware remains a bottleneck. True energy consumption is governed by complex, hardware-specific factors like memory bandwidth and compiler optimizations (e.g., TensorRT fusion), making it far harder to optimize than theoretical FLOPs. Existing HW-NAS methods rely on exhaustive on-device profiling, static Look-Up Tables (LUTs), or few-shot meta-learning. Because all of those require target-device measurements or calibration, true zero-shot cross-hardware energy prediction remains unsolved.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

A parallel bottleneck exists in evaluating model accuracy. To avoid the massive overhead of training candidates, HW-NAS often uses Zero-Cost (ZC) proxies to score untrained networks. However, individual proxies are brittle, and a search that optimizes a single proxy is free to exploit its mathematical blind spots—risking unbalanced "glass cannon" candidates that excel on one theoretical metric without the balanced capacity that practical accuracy requires.

We introduce Energy-Efficiency-Aware Neural Architecture Search (EENAS), a framework that operates entirely zero-shot for both hardware and software. Instead of relying on relative margins, expensive training, or on-device profiling, EENAS utilizes two novel, training-free predictors. This allows practitioners to perform Absolute Hardware Bounding: restricting the search to a strict energy budget on a completely unseen GPU. Our core contributions are as follows:

- **Zero-Shot Cross-Hardware Energy Prediction:** We propose an energy prediction pipeline requiring no target-device calibration. Using a PCA-based anomaly router to dynamically dispatch tasks to either an in-distribution symmetric tree (CatBoost) or an out-of-distribution Tabular Transformer, we achieve $< 10\%$ MAPE on unseen GPUs and architectures.
- **Robust Log Z-Score Accuracy Proxy Aggregation:** We show that gradient-based ZC proxies collapse in heterogeneous search spaces. To guard against single-proxy exploitation, we introduce a Log Z-score aggregation that ensembles multiple ZC proxies at zero additional cost, requiring candidates to balance expressivity, spatial separability, and channel independence.
- **Absolute Hardware Bounding via Offline Search:** Integrating these predictors into an NSGA-II [6] algorithm eliminates the massive GPU-hour footprint of traditional NAS. EENAS evaluates both accuracy and energy via tabular models and mathematical bounds, discovering optimal Pareto fronts for specific budgets entirely offline in under 30 minutes.
- **New Efficiency Frontier:** On ImageNet-1K, EENAS-searched models establish a new energy-accuracy Pareto frontier. Compared to efficient baselines, they use 40% less energy at the same accuracy, or achieve a 2.5% accuracy gain under identical energy constraints.

2 Related Work

Hardware-Aware Neural Architecture Search Hardware-Aware NAS (HW-NAS) integrates physical deployment constraints into the architectural search reward, as seen in MnasNet [32] and the decoupled supernet approach of Once-for-All [3]. Recent frameworks like MO-HDNAS [31] explore multi-objective Pareto fronts across diverse edge environments. However, these methods primarily optimize for inference latency or theoretical FLOPs. While benchmarks like EC-NAS [2] offer tabular datasets for energy costs, end-to-end NAS pipelines that strictly target absolute energy consumption remain underexplored. Existing methods largely rely on exhaustive on-device measurements or static Lookup Tables (LUTs). EENAS bridges this gap by elevating energy consumption to the primary search constraint, enabling absolute hardware bounding without any device-side profiling.

Cross-Device Performance Prediction Surrogate hardware predictors aim to bypass the prohibitive costs of on-device benchmarking. For example, nn-Meter [41] uses random forests for kernel-level latency prediction, while BRP-NAS [7] uses Graph Convolutional Networks (GCNs) to encode architectural topologies. To generalize to unseen devices, methods like HELP [14] and PlatformX [33] employ meta-learning and transferable frameworks, but still require few-shot on-device calibration. EENAS completely eliminates this calibration step, leveraging a PCA-based anomaly router to achieve accurate, strictly zero-shot cross-hardware energy prediction.

Zero-Cost Proxies and Proxy Ensembles Zero-Cost (ZC) proxies accelerate NAS by scoring untrained networks using a single minibatch. Existing proxies capture different properties: NAS-WOT [21] measures activation linear regions, Zen-NAS [19] evaluates perturbation amplification, and NEAR [11] assesses feature rank independence. However, NAS-Bench-Suite-Zero [13] shows that individual proxies are brittle and often disagree across search spaces. We also find that gradient-based proxies such as ZiCo [17] and AZ-NAS [15] can collapse in heterogeneous macro-architectural spaces due to incomparable gradient magnitudes.

Recent methods combine multiple signals to mitigate individual proxy bias. UP-NAS [10] meta-learns a unified multi-proxy estimator, while EZNAS [1] uses genetic programming to evolve aggregation formulas. Unlike these computationally intensive and overfitting-prone meta-learners,

EENAS introduces a training-free Log Z-score aggregation. By ensembling forward-pass proxies geometrically, we require balanced competency across complementary capacity signals at zero computational overhead—inexpensive insurance against single-proxy exploitation.

3 Empirical Motivation and Search Space Design

Profiling SOTA Models To design genuinely energy-efficient models, we must first establish how current state-of-the-art (SOTA) architectures perform under strict energy constraints. We evaluated several prominent model families to establish a baseline:

HGNet-V2 [24], GENet [18], and MobileNetV4 [28] currently define the energy-accuracy Pareto frontier [38]. While HGNet-V2 is hand-designed, GENet and MobileNetV4 employ hardware-aware NAS targeting inference latency rather than energy. GENet optimizes for fixed V100 latency budgets, while MobileNetV4 utilizes latency/cost models for mobile hardware (e.g., EdgeTPU). In both cases, physical hardware measurements are relegated to profiling or final benchmarking, rather than acting as a direct search constraint.

We additionally analyzed GPU-Net [34], an NVIDIA NAS framework optimizing GPU latency; ZenNAS [19] and DeepMAD [30], two zero-shot NAS models targeting latency, parameter, and FLOP counts; and two legacy architectures, VoVNet [16] and HarDNet [4], selected specifically to test their original claims of high energy efficiency and low memory traffic.

Methodology Following the fairness principles established by Yang and Armour [38], we evaluated each model across a sweep of batch sizes, recording the minimum energy consumption per image to account for varying hardware utilization rates (as larger models naturally utilize hardware more efficiently at scale). Measurements were conducted on an NVIDIA RTX 5090 using the widely adopted `nvidia-smi` procedure [40]. To isolate architectural efficiency and eliminate variations induced by resolution scaling, all tests were strictly bounded to a 224×224 input size. Pretrained weights were sourced from the HuggingFace `timm` repository [35].

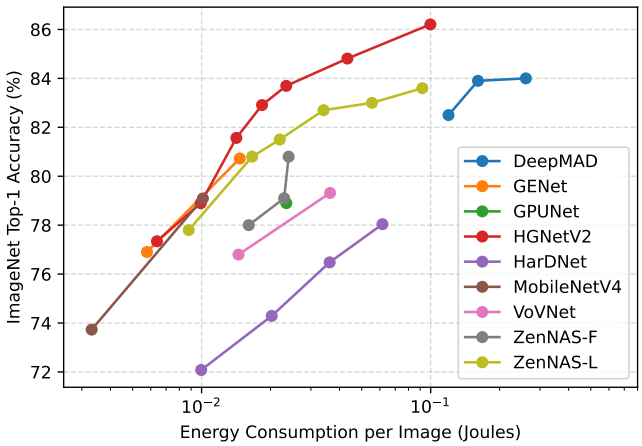

Figure 1: Energy Consumption per Image Inference vs. ImageNet-1K Top-1 Accuracy of selected SOTA models, measured on an RTX 5090.

Fig. 1 illustrates the results. HGNet-V2, GENet, and MobileNetV4 remain the most efficient architectures, with ZenNAS and DeepMAD lagging behind. This confirms that optimizing for proxy metrics like FLOPs or params does not strictly translate to optimal energy efficiency. Hence, we selected the top three model families as our baselines and the structural foundation for our search space.

Search Space Formulation Our search space builds upon the One-Shot Aggregation architecture of PP-HGNetV2 [24], integrated with parameter-efficient design principles from MobileNet [28] and GENet [18]. The macro-architecture features a specialized multi-layer stem followed by four computational stages. Within these stages, we independently search for optimal depths and monotonically increasing channel widths. At the micro-architectural level, the space explores optimal

spatial receptive fields by mixing convolution kernel sizes, alongside varying Squeeze-and-Excitation (SE) reduction factors and the optional deployment of Learnable Affine Blocks (LAB) for feature recalibration. To manage the parameter count and energy footprint at higher tensor depths, the search space permits the structural substitution of standard dense convolutions with lightweight point-wise and depth-wise block operations.

Concretely, each architecture is encoded as a 16-dimensional genotype: eight channel-width genes specifying the input, intermediate, and output widths of the four stages (drawn from 14 discrete widths between 16 and 2048), and eight structural genes (per-stage block counts, the lightweight-block deployment pattern, the stage-wise kernel schedule, the per-block layer count, LAB on/off, and the SE reduction factor). Channel genes are subject to validity constraints—monotonically increasing widths through the network, a per-stage expansion ratio of at most $8\times$, a first-stage input width of at most 96, and a final-to-first-stage output-width ratio within $[8\times, 16\times]$—whose exact enumeration yields 20,810 valid channel configurations. Combined with 16,200 structural combinations per configuration, the space contains $\approx 3.4 \times 10^8$ valid architectures. Appendix F specifies every variable, its allowed values, all constraints, and the genotype-to-network mapping.

4 The Zero-Shot Energy Consumption Surrogate

4.1 Dataset Generation and Feature Representation

To train a cross-GPU energy predictor capable of zero-shot generalization, we first constructed a hardware-benchmarked dataset that captures the non-linear relationship between architectural topology and hardware execution.

Architecture Sampling and Fair Measurement We sampled 2,000 unique network architectures from our search space using Latin Hypercube Sampling (LHS). This minimizes random sampling bias and ensures uniform coverage across the highly multi-dimensional bounds of the space.

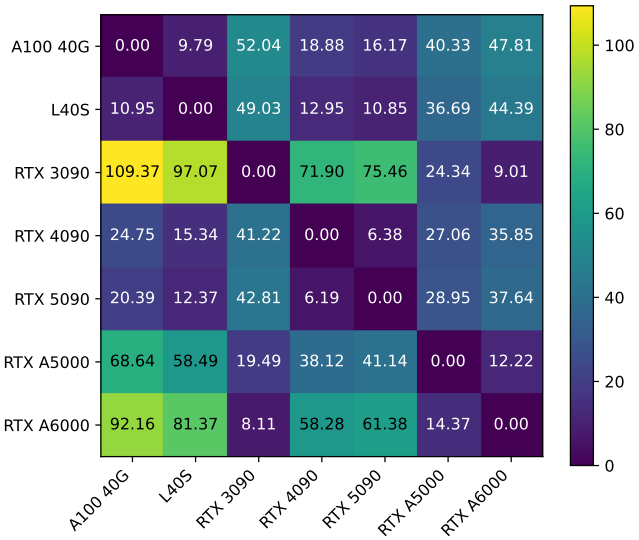

Figure 2: Mean Absolute Percentage Error (MAPE) of the energy consumption of the 2,000 sampled architectures on the 7 different GPUs.

We evaluated all 2,000 architectures across seven NVIDIA GPUs: RTX 3090, RTX 4090, RTX 5090, RTX A5000, RTX A6000, L40S, and A100 (40GB). This suite spans consumer, workstation, and datacenter tiers across the Ampere, Ada Lovelace, and Blackwell architectures. Following the fair comparison principles discussed in Section 3, we tested each model across a sweep of batch sizes. The minimum recorded energy consumption establishes the ground-truth energy cost, $E(a, g)$, for network architecture a on GPU g. Appendix J accounts for the one-time energy cost of this measurement campaign and its amortisation, together with the cost of the in-loop-measurement

alternative that the predictor replaces; Appendix I shows empirically why cheaper measurement regimes cannot substitute for this protocol.

Feature Vector Formulation We formulate the predictive task by mapping each architecture-hardware pair into a joint feature vector $x = [x_{arch}, x_{hw}]$:

- **Architecture Features (x_{arch}):** A 19-dimensional vector comprising the 16 genotype variables that define the macro- and micro-architecture (e.g., channel widths, stage depths, kernel combinations), appended with three offline-calculated graph metrics: parameter count, FLOPs, and total activations.
- **Hardware Features (x_{hw}):** A dense vector detailing the target GPU's physical and architectural specifications. Key metrics include Thermal Design Power (TDP), core and memory clock speeds, memory bandwidth, CUDA and Tensor core counts, and theoretical peak FP16/FP32 performance (see Appendix A for the complete feature list).

This joint representation provides downstream predictors with the necessary context to model both the structural computational complexity of the graph and the specific bandwidth and compute bottlenecks of the target hardware.

4.2 The Generalization Dichotomy

4.2.1 In-Distribution Prediction via Symmetric Gradient Boosting

Justification for Tree-Based Models The joint feature space $x = [x_{arch}, x_{hw}]$ casts cross-device energy prediction as a tabular regression problem with highly heterogeneous distributions, as architectural topologies and hardware metrics operate on vastly different scales.

While deep neural networks excel on unstructured data, Gradient Boosted Decision Trees (GBDTs) offer a stronger inductive bias for unaligned tabular datasets. GBDTs are natively invariant to monotonic feature transformations and effectively capture sharp, non-linear threshold effects.

We select CatBoost [27] as our foundational predictor. Unlike XGBoost [5] or LightGBM [12], which grow trees asymmetrically, CatBoost builds oblivious (symmetric) decision trees that apply the same splitting criterion across an entire level. For hardware-aware NAS, this symmetry provides strong structural regularization. It forces the model to learn universal feature interactions—such as the balance between parameter count and memory bandwidth—rather than growing isolated, deep branches that overfit to a single training GPU's specific bottlenecks.

Target Transformation and Regularization Strategy Because energy consumption $E(a, g)$ spans several orders of magnitude, minimizing loss directly in linear space (Joules) disproportionately penalizes errors on larger models. To stabilize variance and preserve relative architectural ranking, we transform the target variable into logarithmic space.

To prevent overfitting, we restrict the maximum tree depth to 6 and apply strong L2 leaf regularization ($\lambda = 10$). The model is optimized via Root Mean Squared Error (RMSE) for 6,000 iterations with a conservative learning rate ($\alpha = 0.03$), utilizing early stopping on an unseen-architecture validation set to halt training when generalization plateaus.

The Quadrant Evaluation Strategy To evaluate generalization boundaries, we partition the dataset into a four-quadrant matrix by crossing architecture novelty with hardware novelty: the training set, Hardware Transfer (seen architectures on unseen GPUs), Software Transfer (unseen architectures on seen GPUs), and the Zero-Shot test quadrant (unseen architectures on unseen GPUs); Appendix D details the protocol.

While the regularized CatBoost model performs robustly on Software Transfer, empirical evaluation reveals an inductive limitation: symmetric trees struggle to extrapolate under extreme Hardware Transfer (e.g., predicting energy on an unseen Blackwell RTX 5090 from Ampere-generation data alone). This bounded extrapolation necessitates a secondary, out-of-distribution predictor.

4.2.2 Out-of-Distribution Extrapolation via Tabular Transformers

While symmetric decision trees excel at mapping in-distribution hardware profiles, their orthogonal feature partitioning prevents them from extrapolating continuous scaling laws. When queried with out-of-distribution (OOD) targets, such as generalizing from consumer-grade GPUs to datacenter-tier

A100s, tree leaves remain constant beyond their training support, leading to bounded extrapolation. To resolve zero-shot hardware transfer, we introduce a custom model for continuous tabular extrapolation.

Feature Tokenization and Architecture Standard Multi-Layer Perceptrons (MLPs) often overfit on tabular data. To enforce structured representation learning, we adopt a Feature-Tokenizer Transformer (FT-Transformer) approach [9]. Instead of processing the joint feature vector $x \in \mathbb{R}^k$ as a flat input, we project each scalar feature x_i into a dense d-dimensional embedding $e_i \in \mathbb{R}^d$: $e_i = w_i x_i + b_i$

We prepend a learnable [CLS] token to this embedding sequence, which is processed by a highly constrained, shallow self-attention encoder (2 layers, 4 attention heads, $d = 24$). This compact dimensionality is deliberate; it restricts network capacity to prevent memorization of training hardware profiles and forces the model to learn generalized cross-attention between software constraints and hardware bottlenecks. The final energy prediction is projected via an MLP head.

Training Dynamics and Regularization We employ bounded manifold mixup and feature masking to prevent attention collapse and enforce linear transitions in the OOD space (see Appendix C for full training dynamics).

Custom Loss and Deep Ensembling While the model targets logarithmic energy $\log_{10}(E)$, optimizing in log-space can distort true error magnitudes. We instead use a differentiable Mean Absolute Percentage Error (MAPE) loss computed in linear space:

$$\mathcal{L} = \frac{1}{B} \sum_{i=1}^{B} \left| \frac{10^{\hat{y}_i} - 10^{y_i}}{10^{y_i} + \epsilon} \right|$$

Optimization utilizes AdamW with weight decay ($\lambda = 10^{-2}$), gradient clipping, and a Cosine Annealing schedule with a 10% linear warmup. Finally, to smooth the prediction landscape and reduce variance, we deploy a Deep Ensemble of five independent models with varied initializations, averaging their linear-space predictions for the final OOD result.

4.3 Dynamic Model Routing via PCA Reconstruction Error

To leverage the complementary strengths of symmetric trees and transformers, we implement a dynamic routing mechanism that evaluates a target GPU g_{test} to select the optimal predictor.

PCA Reconstruction for Anomaly Detection We detect anomalies by subspace projection via Principal Component Analysis (PCA) reconstruction error. Restricting PCA to $k < N_{train}$ components captures the dominant structure the known devices share, such as the relationship between core counts and memory bandwidth, and reconstruction error measures a target's departure from it; we use $k = 4$, which retains 99% of the hardware-feature variance.

Let $X_{train} \in \mathbb{R}^{N \times p}$ be the standardized hardware features of the known GPUs. We compute the principal component projection matrix $W \in \mathbb{R}^{p \times k}$. For an unseen target GPU x_{test}, we compute its reconstruction $\hat{x}_{test} = (x_{test}W)W^T$.

The anomaly score is the Mean Squared Error (MSE) of this reconstruction:

$$E_{recon}(x_{test}) = \frac{1}{p} \sum_{i=1}^{p} (x_{test,i} - \hat{x}_{test,i})^2$$

If the unseen GPU aligns with familiar hardware, its feature ratios will yield a low E_{recon}. Conversely, a novel hardware paradigm or bottleneck will result in a significantly higher reconstruction error.

Non-Parametric Thresholding and Routing Logic Selecting an anomaly threshold τ in a micro-dataset ($N_{train} = 6$) requires statistical robustness. Standard parametric thresholds, such as the Z-score ($\mu + z\sigma$), are unsuitable here; a single OOD entry (e.g., the A100) can inflate both the sample mean and variance, masking its own anomalous nature.

We ground our threshold in non-parametric statistics using the median cross-validation reconstruction error, $\tilde{x}_E$, which has a high breakdown point and remains immune to extreme spikes. Because MSE is strictly positive and right-skewed in hardware spaces, we apply a multiplicative bound $\tau = 2 \times \tilde{x}_E$.

This scale-invariant tolerance asserts that a target hardware represents a paradigm shift if its structural divergence is more than double the typical variance observed in the known pool:

$$\text{Predictor} = \begin{cases} f_{\text{CatBoost}} & \text{if } E_{recon}(x_{test}) \leq 2 \times \tilde{x}_E \\ f_{\text{Transformer}} & \text{if } E_{recon}(x_{test}) > 2 \times \tilde{x}_E \end{cases}$$

During evaluation, this bound correctly identified similar hardware (e.g., RTX 3090 to A6000) as in-distribution, with errors near the median ($\sim 0.06 \sim 0.08$). In contrast, the High Bandwidth Memory of the A100 (0.407) and the new Blackwell architecture of the RTX 5090 (0.204) violated the bound, appropriately triggering the robust Transformer ensemble. The routing is furthermore insensitive to the exact multiplier: sweeping $m \in \{1.0, \ldots, 4.0\}$ in $\tau = m \times \tilde{x}_E$ leaves all seven leave-one-out routing decisions unchanged throughout $m \in [1.25, 2.5]$, so the chosen $m = 2$ lies inside a stable region rather than near a decision boundary (Appendix G).

4.4 Zero-Shot Energy Prediction Results

We evaluate the energy prediction pipeline in a leave-one-GPU-out setting. For each target GPU in turn, every energy measurement taken on that GPU is withheld: the predictors are trained only on the remaining six GPUs, early stopping is supervised only by unseen architectures on those six GPUs (Appendix D), and no hyperparameter is tuned against the target. The target's hardware specifications are the only information about it that the pipeline consults, exactly as in deployment. Each reported result is then computed on the 400 held-out architectures measured on the withheld GPU—architectures never seen during training on any GPU—so the numbers in Table 1 reflect simultaneous transfer to unseen hardware and unseen architectures. As the table shows, relying on a single predictive model leads to significant failure modes under hardware distribution shifts.

The Failure of Standalone Predictors The symmetric decision tree (CatBoost) achieves high precision on in-distribution generational transfers—such as 4.80% MAPE on the L40S and 6.25% on the RTX 4090—but fails to extrapolate to architectural anomalies. When tested on the High Bandwidth Memory of the datacenter A100 or the Blackwell-based RTX 5090, CatBoost performance degrades significantly, with MAPE values reaching 41.59% and 26.53%, respectively.

Conversely, the Tabular Transformer ensemble handles these OOD shifts well, maintaining error rates below 10% for the A100 and RTX 5090. However, this flexibility sacrifices local precision on familiar hardware; the network suffers >20% MAPE on standard workstation cards like the RTX A5000 and A6000, likely due to its inability to capture the sharp, non-linear thresholds of more standard hardware.

Table 1: Zero-Shot Energy Prediction Performance on Unseen Hardware. Bold: predictor selected by the router.

			Predictor Configuration (MAPE / Spearman)		
Target GPU	E_{recon}	Anomaly	CatBoost Only	Transformer Only	**EENAS (Routed)**
A100 40GB	0.407	Outlier	41.59% / 0.992	**7.97% / 0.988**	**7.97% / 0.988**
RTX 3090	0.085	Inlier	**7.71% / 0.998**	6.99% / 0.996	**7.71% / 0.998**
RTX 4090	0.056	Inlier	**6.25% / 0.998**	15.81% / 0.997	**6.25% / 0.998**
RTX 5090	0.204	Outlier	26.53% / 0.997	**9.93% / 0.997**	**9.93% / 0.997**
L40S	0.081	Inlier	**4.80% / 0.996**	19.71% / 0.996	**4.80% / 0.996**
RTX A5000	0.061	Inlier	**9.86% / 0.996**	22.27% / 0.997	**9.86% / 0.996**
RTX A6000	0.068	Inlier	**9.69% / 0.997**	22.43% / 0.997	**9.69% / 0.997**

The Success of the EENAS Router The PCA-based dynamic routing mechanism resolves this performance gap. By identifying the A100 and RTX 5090 as statistical outliers based on reconstruction error, the router dispatches these cases to the Transformer ensemble while maintaining high-precision CatBoost routing for standard hardware.

This unified EENAS predictor bounds zero-shot MAPE to $\sim 10\%$ or lower across the entire heterogeneous GPU suite. Crucially for NAS, the routed predictor maintains an exceptional Spearman rank correlation (above 0.98), ensuring that the relative energy efficiency ordering of candidate

Table 2: Rank correlation between zero-cost proxy scores and the target ranking. Zen achieves the strongest correlation, followed by NASWOT and NEAR, while AZ_TRA shows a negative correlation.

Proxy	Spearman ρ	Kendall τ
AZ_EXP [15]	0.0972	0.0631
AZ_TRA [15]	-0.0587	-0.0401
EPE [20]	0.2601	0.1697
NASWOT [21]	**0.8084**	**0.6095**
NEAR [11]	**0.7149**	**0.5328**
Zen [19]	**0.8747**	**0.7195**
ZiCo [17]	0.0947	0.0622

architectures is preserved regardless of hardware constraints. Since both predictors preserve ranking almost perfectly, the router's contribution is absolute calibration—precisely the property that Absolute Hardware Bounding (Section 6) consumes; Appendix H quantifies the cost of miscalibrated bounding at the selection level and compares against few-shot on-device calibration at matched budgets.

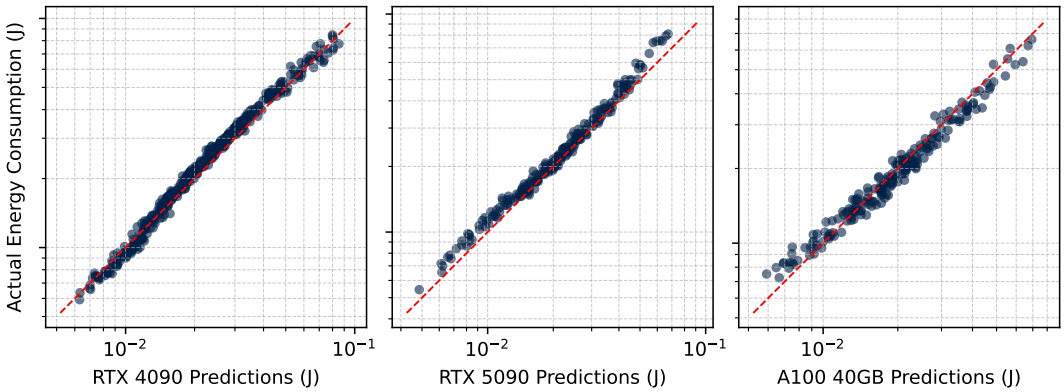

Figure 3: Predicted v.s. Actual model energy consumption on RTX 4090, RTX 5090 and A100 40GB.

5 Robust Zero-Shot Accuracy Proxy Ensembles

Zero-Cost (ZC) proxies evaluate the theoretical potential of an untrained network $f(x;\theta_0)$ using a single minibatch of data. However, ZC proxy effectiveness is highly search-space dependent. Relying on a single proxy also leaves evolutionary search free to exploit that proxy's algorithmic blind spots, risking candidates that maximize one theoretical metric without balanced capacity across the properties practical accuracy depends on.

Empirical Proxy Selection To identify optimal predictive signals for our search space, we uniformly sampled 100 architectures and trained them for 12 epochs on ImageNet-1K to establish a ground-truth accuracy ranking. We then evaluated seven state-of-the-art ZC proxies against this benchmark.

As shown in Table 2, gradient- and Jacobian-based trainability proxies (ZiCo, AZ-NAS) collapse below $\rho = 0.10$ in our space—typical of heterogeneous macro-architectures, where mixing dense and depthwise blocks makes raw gradient magnitudes incomparable across models—whereas three capacity metrics correlate strongly: Zen-Score (perturbation amplification, $\rho = 0.87$), NASWOT (spatial linear separability, $\rho = 0.81$), and NEAR (activation-rank expressivity, $\rho = 0.71$).

Log Z-Score Aggregation While Zen, NASWOT, and NEAR each provide a strong signal, their inter-proxy rank correlation over our 2,000-sample pool is only 0.70–0.80: a candidate can rank highly on one capacity axis while lagging on another. We therefore combine them with a simple, training-free aggregation: each proxy $p_i \in \{\text{Zen, NASWOT, NEAR}\}$ is normalized to a z-score $z_i(a) = (p_i(a) - \mu_i)/\sigma_i$ using population statistics of the 2,000-sample pool, and the composite

sums the logarithms of the shifted z-scores, where K ensures strict positivity:

$$S_{accuracy}(a) = \sum_{i \in \{\text{Zen, NASWOT, NEAR}\}} \log(z_i(a) + K)$$

The logarithm down-weights candidates with a pronounced deficit on any single axis, encouraging balanced competency across expressivity, spatial separability, and channel independence. The composite matches the standalone Zen proxy's fidelity ($\rho = 0.86$ on the 100-sample subset) while remaining free of fragile hyperparameters—its ranking is flat over $K \in [2, 12]$ and stable under 200 random proxy weightings—and it transfers to full training length, ordering the seven fully trained (300-epoch) EENAS models perfectly ($\rho = \tau = 1.0$; Appendix G). Since the energy objective already bounds the most direct exploitation route (inflating capacity-correlated scores through sheer scale), the aggregation acts as a zero-cost balance requirement complementing that pressure.

6 End-to-End Architecture Search & Final Results

6.1 Fast Evolutionary Search via Zero-Shot Predictors

Equipped with our cross-GPU energy predictor and robust accuracy proxy ensemble, the HW-NAS problem reduces to an offline optimization task. We utilize the Non-Dominated Sorting Genetic Algorithm (NSGA-II) to navigate the discrete topology of the search space.

Absolute Hardware Bounding via Zero-Shot Prediction Because our routed energy predictor maintains a zero-shot MAPE below 10%, we can introduce Absolute Hardware Bounding. Rather than exploring a generic Pareto front, practitioners can define a strict energy budget C_{max} for an unbenchmarked target GPU. Any candidate that exceeds this prediction bound is immediately discarded. This tightly confines the search to the viable operational envelope of the target hardware. Because MAPE is a mean rather than a worst-case bound, deployments with strict caps can additionally retain only candidates satisfying $\hat{E}(a, g) \leq 0.9\, C_{max}$, which guarantees $E \leq C_{max}$ whenever an individual under-prediction does not exceed 10%; the small Pareto set of finalists is then physically validated on the target before deployment. Appendix H analyses bounding under prediction uncertainty in detail.

Multi-Objective Optimization Pipeline The evolutionary search initializes with a uniformly sampled population of $N = 200$ architectures. Over $G = 64$ generations, the algorithm seeks the Pareto-optimal set of architectures a^* that maximizes our aggregated accuracy proxy $S_{accuracy}(a)$ while minimizing predicted energy consumption:

$$a^* \in \underset{a \in \mathcal{A}}{\text{argmin}} \begin{bmatrix} -S_{accuracy}(a) \\ \hat{E}(a, g_{target}) \end{bmatrix} \quad \text{s.t.} \quad \hat{E}(a, g_{target}) \leq C_{max}$$

To generate offspring, we select the top 50 non-dominated elites from each generation based on crowding distance. We apply a hybrid generation strategy using discrete categorical mutations, block-level crossovers, stage-group crossovers, and uniform random injections to maintain global exploration and prevent convergence in local minima.

Sub-30 Minute Evaluation Cache Evaluating a single generation in standard HW-NAS often requires on-device latency measurements or partial network training, consuming hundreds of GPU hours. In contrast, EENAS evaluates candidates exclusively via forward-pass mathematical proxies and tabular predictors. We further accelerate this process by implementing a persistent cache that hashes the 16-dimensional architectural genome. Because the predictive models are lightweight and duplicate evaluations are cached, the entire 64-generation multi-objective search—evaluating thousands of networks—completes offline in under 30 minutes on a single consumer workstation. Appendix K verifies that the searched candidates dominate the raw sampling pool of the same space at every energy budget, attributing the final gains to the search rather than the search space alone.

6.2 Final Results

Standard Training Recipe To isolate architectural quality, all baseline and candidate models are trained from scratch using a standard recipe [35] (refer to Appendix B for more detail).

Results on ImageNet-1K We evaluate the EENAS-discovered architectures targeting the NVIDIA RTX 5090. Following the offline search, we physically validate the true energy consumption of the selected models on the target hardware.

The EENAS-searched models successfully establish a new energy-accuracy Pareto frontier. At an identical energy budget, our architectures achieve a +2.5% improvement in ImageNet Top-1 accuracy over the most efficient baselines. Conversely, when matching baseline accuracy, EENAS models reduce absolute energy consumption by approximately 40%. These empirical gains validate that explicitly constraining architecture search via strict, zero-shot energy prediction yields fundamentally more efficient topologies than optimizing for latency or FLOPs alone.

We emphasize the comparison scope: Fig. 4 compares *architectures* under one identical training investment, whereas publicly released checkpoints (Fig. 1) embed per-model training pipelines with substantially larger budgets—for instance, the published MobileNetV4-Conv-S result uses a 9,600-epoch schedule versus our unified 300-epoch budget, and training-recipe effects alone can exceed the resulting accuracy gaps [36]. Such investments are architecture-agnostic and would shift all operating points upward together. Appendix L details the comparability scope and the published baselines' training configurations.

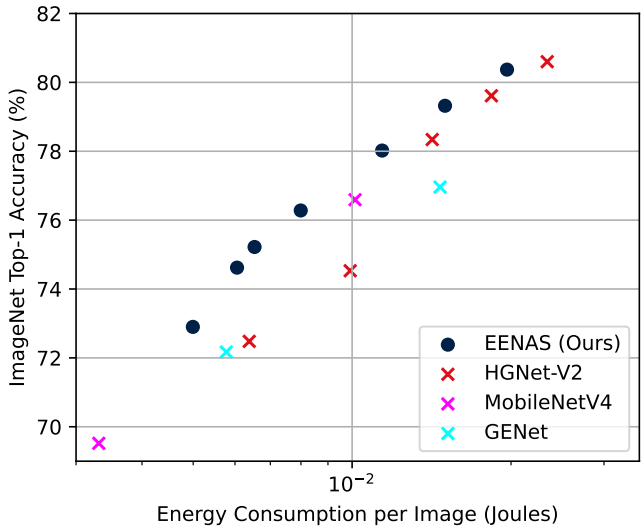

Figure 4: Energy v.s. Accuracy of EENAS (ours) and baseline models trained on the standard recipe.

Limitations Our cross-hardware claims are established for NVIDIA GPUs (seven devices spanning three architecture generations); extending the hardware encoding to other accelerator classes (TPUs, mobile NPUs) requires vendor-neutral features and representative training hardware, and is not demonstrated here. The PCA routing threshold and the proxy selection are fitted on small samples and should be refitted when the hardware pool or search space changes materially. Accuracy proxies are validated against 12-epoch training on 100 architectures, with full-training consistency confirmed on the seven fully trained models ($\rho = 1.0$, $n = 7$). Energy is measured via the GPUs' built-in sensors following the validated procedure of Yang et al. [40], using steady-state batch-size sweeps; deployed finalists are additionally physically validated on the target.

7 Conclusion

We introduce Energy-Efficiency-Aware Neural Architecture Search (EENAS), a strictly zero-shot framework that eliminates the need for physical hardware calibration. EENAS utilizes a PCA-based anomaly router for highly accurate (<10% MAPE) cross-hardware energy prediction, alongside a training-free Log Z-score aggregation of multiple proxies to robustly estimate accuracy. This dual zero-shot approach enables Absolute Hardware Bounding, discovering Pareto-optimal architectures entirely offline in under 30 minutes. Evaluated on ImageNet-1K, EENAS delivers models consuming 40% less energy or achieving 2.5% higher accuracy than highly efficient baselines.

Acknowledgments and Disclosure of Funding

The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work [29]. This work received no specific funding. The authors declare no competing interests.

References

[1] Yash Akhauri, Juan Munoz, Nilesh Jain, and Ravishankar Iyer. Eznas: evolving zero-cost proxies for neural architecture scoring. *Advances in Neural Information Processing Systems*, 35:30459–30470, 2022.

[2] Pedram Bakhtiarifard, Christian Igel, and Raghavendra Selvan. Ec-nas: Energy consumption aware tabular benchmarks for neural architecture search. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5660–5664. IEEE, 2024.

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HylxE1HKwS`.

[4] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[5] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[7] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. *Advances in neural information processing systems*, 33:10480–10490, 2020.

[8] Google. Google environment report 2025. `https://www.gstatic.com/gumdrop/sustainability/google-2025-environmental-report.pdf`, 2025.

[9] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34: 18932–18943, 2021.

[10] Yi-Cheng Huang, Wei-Hua Li, Chih-Han Tsou, Jun-Cheng Chen, and Chu-Song Chen. Up-nas: Unified proxy for neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1675–1684, 2024.

[11] Raphael T Husistein, Markus Reiher, and Marco Eckhoff. Near: A training-free pre-estimator of machine learning model performance. *arXiv preprint arXiv:2408.08776*, 2024.

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[13] Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. Nas-bench-suite-zero: Accelerating research on zero cost proxies. *Advances in Neural Information Processing Systems*, 35:28037–28051, 2022.

[14] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. Hardware-adaptive efficient latency prediction for nas via meta-learning. *Advances in Neural Information Processing Systems*, 34:27016–27028, 2021.

[15] Junghyup Lee and Bumsub Ham. Az-nas: Assembling zero-cost proxies for network architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5893–5903, 2024.

[16] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[17] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023.

[18] Ming Lin, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Neural architecture design for gpu-efficient networks. *arXiv preprint arXiv:2006.14090*, 2020.

[19] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance image recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 347–356, 2021.

[20] Vasco Lopes, Saeid Alirezazadeh, and Luís A Alexandre. Epe-nas: Efficient performance estimation without training for neural architecture search. In *International conference on artificial neural networks*, pages 552–563. Springer, 2021.

[21] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International conference on machine learning*, pages 7588–7598. PMLR, 2021.

[22] Meta. Meta 2025 environmental data index. `https://sustainability.atmeta.com/wp-content/uploads/2025/10/Meta_2025-Environmental-Data-Index.pdf`, 2025.

[23] Microsoft. 2025 environmental sustainability report. `https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/msc/documents/presentations/CSR/2025-Microsoft-Environmental-Sustainability-Report-PDF.pdf`, 2025.

[24] PaddlePaddle. Pp-hgnetv2 model documentation, October 2023. URL `https://github.com/PaddlePaddle/PaddleClas/blob/develop/docs/zh_CN/models/ImageNet1k/PP-HGNetV2.md`.

[25] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28, 2022.

[26] Lucas Caetano Pereira, Bruna Guterres, Kauê Sbrissa, Amanda Mendes, Francisca Vermeulen, Lisl Lain, Marié Smith, Javier Martinez, Paulo Drews, Nelson Duarte, et al. The not-so-easy task of taking heavy-lift ml models to the edge: A performance-watt perspective. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 699–706, 2023.

[27] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[28] Danfeng Qin, Chas Leichner, Manolis Delakis, Marco Fornoni, Shixin Luo, Fan Yang, Weijun Wang, Colby Banbury, Chengxi Ye, Berkin Akin, et al. Mobilenetv4: Universal models for the mobile ecosystem. In *European Conference on Computer Vision*, pages 78–96. Springer, 2025.

[29] Andrew Richards. University of oxford advanced research computing, 2015. URL `https://doi.org/10.5281/zenodo.22558`.

[30] Xuan Shen, Yaohua Wang, Ming Lin, Yilun Huang, Hao Tang, Xiuyu Sun, and Yanzhi Wang. Deepmad: Mathematical architecture design for deep convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. URL `https://arxiv.org/abs/2303.02165`.

[31] Nilotpal Sinha, Peyman Rostami, Abd El Rahman Shabayek, Anis Kacem, and Djamila Aouada. Multi-objective hardware aware neural architecture search using hardware cost diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8032–8039, 2024.

[32] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[33] Xiaolong Tu, Dawei Chen, Kyungtae Han, Onur Altintas, and Haoxin Wang. Platformx: An end-to-end transferable platform for energy-efficient neural architecture search. In *Proceedings of the Tenth ACM/IEEE Symposium on Edge Computing*, pages 1–16, 2025.

[34] Linnan Wang, Chenhan Yu, Satish Salian, Slawomir Kierat, Szymon Migacz, and Alex Fit Florea. Searching the deployable convolution neural networks for gpus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12227–12236, 2022.

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

[36] Ross Wightman, Hugo Touvron, and Hervé Jégou. ResNet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

[37] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[38] Zeyu Yang and Wesley Armour. The hidden joules: Evaluating the energy consumption of vision backbones for progress towards more efficient model inference. In *Forty-second International Conference on Machine Learning*, 2025.

[39] Zeyu Yang, Angus B Clark, Digby Chappell, and Nicolas Rojas. Instinctive real-time semg-based control of prosthetic hand with reduced data acquisition and embedded deep learning training. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5666–5672. IEEE, 2022.

[40] Zeyu Yang, Karel Adamek, and Wesley Armour. Accurate and convenient energy measurements for gpus: A detailed study of nvidia gpu's built-in power sensor. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–17. IEEE, 2024.

[41] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. Nn-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–93, 2021.

[42] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=r1Ue8Hcxg`.

A Hardware Features

Table 3 describes the GPU's physical and architectural envelope: thermal design power, core and memory clocks, memory bandwidth, CUDA- and Tensor-core counts, cache and memory capacities, and theoretical peak FP16/FP32 throughput. Together they give a predictor the context to model both the structural complexity of the network and the compute and bandwidth bottlenecks of the device executing it.

Table 3: GPU hardware features used by the energy predictor.

Category	Feature	Description
Power	`tdp_W`	Thermal design power in watts
Compute resources	`sm_count`	Number of streaming multiprocessors
	`cuda_cores`	Number of CUDA cores
	`tensor_cores`	Number of tensor cores
	`sm_clock_MHz`	Streaming multiprocessor clock frequency
Peak compute	`cuda_core_pk_32bit_GFLOPS`	Peak FP32 CUDA-core throughput
	`cuda_core_pk_16bit_GFLOPS`	Peak FP16 CUDA-core throughput
	`tensor_core_pk_32bit_GFLOPS`	Peak tensor-core FP32/TF32 throughput
	`tensor_core_pk_16bit_GFLOPS`	Peak tensor-core FP16/BF16 throughput
Memory and cache	`l2_cache_size_MB`	L2 cache size
	`total_vram_GB`	Total GPU memory capacity
	`mem_clock_MHz`	Memory clock frequency
	`bus_width_bits`	Memory bus width
	`mem_bw_MBps`	Theoretical memory bandwidth

B Standard Training Recipe

All baseline and searched architectures are trained on ImageNet-1K at 224×224 resolution using the same standardized recipe to ensure a fair comparison. We train for 300 epochs with AdamW, a cosine learning-rate schedule, an initial learning rate of 10^{-3}, weight decay of 0.05, and a 15-epoch warmup period. Training uses distributed data parallelism with a fixed per-GPU batch size, WebDataset-based ImageNet loading, pinned memory, 20 data-loading workers, channels-last memory layout, and PyTorch compilation with the Inductor backend to improve throughput. Mixed-precision training is performed with bfloat16 automatic mixed precision. To improve generalization, we apply a strong augmentation and regularization pipeline consisting of RandAugment, random resized interpolation, Mixup with $\alpha = 0.8$, CutMix with $\alpha = 1.0$, random erasing with probability 0.25, binary cross-entropy loss, stochastic depth with drop-path rate 0.1, gradient clipping at 5.0, and exponential moving average of model weights with decay 0.9998. This recipe is used unchanged across both baseline models and EENAS-discovered architectures, so that differences in accuracy and measured energy consumption primarily reflect architectural differences rather than training-procedure variation.

C FT Transformer Training Dynamics and Regularization

To handle heavy-tailed tabular distributions, we apply $\log_{10}$ transformations to magnitude-varying features (e.g., FLOPs, activations, VRAM, and memory bandwidth) before global standardization. To bridge the interpolation gap between hardware clusters in the feature space, we employ a specific regularization strategy:

- Bounded Manifold Mixup: With probability $p = 0.20$, we interpolate between random architecture/GPU pairs: $\tilde{x} = \lambda x_i + (1 - \lambda)x_j$, where $\lambda \sim \text{Beta}(0.2, 0.2)$. This forces the model to learn linear transitions in the OOD space. To ensure convergence on the true data manifold, Mixup is faded out during the final 20% of training.
- Feature Masking: We stochastically zero out entire input features to prevent the attention mechanism from collapsing onto a single dominant metric, such as peak theoretical FLOPS.

D Quadrant Evaluation Protocol

To evaluate the generalization boundaries of each predictor, we partition the measured dataset into four quadrants, crossing architecture novelty with hardware novelty:

1. Training Set: Seen architectures on seen GPUs.
2. Hardware Transfer (Val A): Seen architectures evaluated on entirely unseen GPUs.
3. Software Transfer (Val B): Unseen architectures evaluated on seen GPUs.
4. Zero-Shot (Test): Unseen architectures evaluated on unseen GPUs.

Of the 2,000 sampled architectures, 1,600 serve as seen and 400 are held out as unseen. The training quadrant fits the predictors and the Software-Transfer quadrant supervises early stopping; all zero-shot results (Table 1) are computed exclusively on the fourth quadrant under leave-one-GPU-out cross-validation, so neither the target GPU nor the test architectures influence training in any way.

E ZC Proxy Scores vs. Test Training Accuracy

Fig. 5 shows the Zen, NASWOT, and NEAR proxy scores versus 12-epoch ImageNet top-1 accuracy over the 100-architecture validation set.

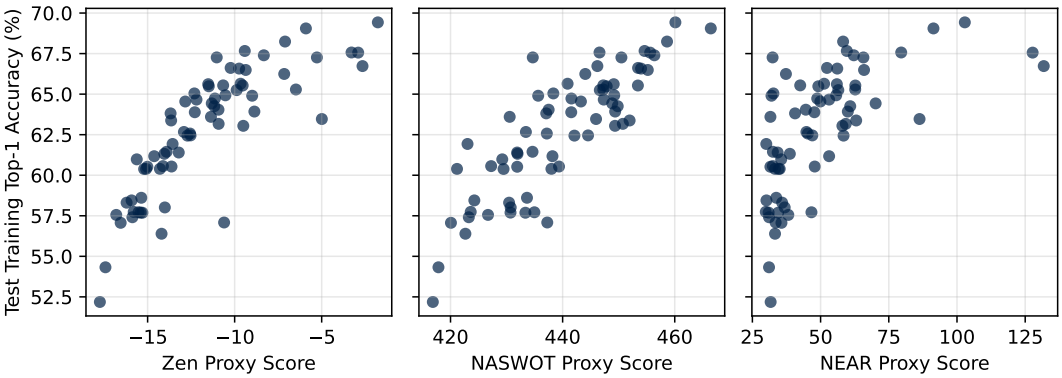

Figure 5: Zen, NASWOT, and NEAR proxy scores v.s. the 12-epoch test training accuracy.

F Complete Search-Space Specification

Each architecture is a 16-dimensional genotype $g_0, \ldots, g_{15}$. Table 4 lists every variable and its allowed values; the constraints and derived quantities follow.

Table 4: Search-space variables. Genes g_0–g_7 are channel widths; g_8–g_{15} are structural choices.

Gene(s)	Variable	Allowed values
g_0	Stage-1 input/intermediate width	$\{16, 32, 48, 64, 96\}$ (constrained to ≤ 96)
g_1–g_7	Stage in/intermediate/out widths	$\{16, 32, 48, 64, 96, 128, 192,$ $256, 384, 512, 768, 1024, 1536, 2048\}$
g_8	Stage-1 block count	$\{1, 2\}$
g_9	Stage-2/4 (shared) block count	$\{1, 2, 3\}$
g_{10}	Stage-3 block count	$\{1, 2, 3, 4, 5, 6\}$
g_{11}	Lightweight-block pattern (stages 1–4)	{[F,T,T,T], [F,F,T,T], [F,F,F,T]}
g_{12}	Kernel-size schedule (stages 1–4)	$\{[3, 3, 3, 5], [3, 3, 5, 5], [3, 5, 5, 5]\}$
g_{13}	Layers per aggregation block	$\{2, 3, 4, 5, 6\}$
g_{14}	Learnable Affine Blocks	{on, off}
g_{15}	SE reduction factor	$\{0.25, 0.375, 0.5, 0.625, 0.75\}$

Channel-validity constraints (i) widths increase monotonically through the network: $g_1 > g_0$, $g_3 > g_1 \geq g_2$, $g_5 > g_3 \geq g_4$, $g_7 > g_5 \geq g_6$, and intermediate widths increase across stages

($g_0 < g_2 < g_4 < g_6$); (ii) each stage's expansion ratio is at most $8\times$; (iii) the first-stage width satisfies $g_0 \leq 96$; (iv) the final-to-first-stage output ratio satisfies $8 \leq g_7/g_1 \leq 16$. Exact enumeration under these constraints yields **20,810** valid channel configurations. The structural genes contribute $2 \times 3 \times 6 \times 3 \times 3 \times 5 \times 2 \times 5 = 16{,}200$ combinations, for a total of $20{,}810 \times 16{,}200 \approx 3.37 \times 10^8$ valid architectures.

Derived quantities The stem's intermediate width follows the first-stage width via the fixed map $\{16{:}16,\ 32{:}24,\ 48{:}32,\ 64{:}32,\ 96{:}48\}$; the classifier-head hidden width is 1024 if the final-stage output width is below 512 and 2048 otherwise; lightweight blocks and kernel sizes follow the stage-wise patterns of g_{11}/g_{12} rather than independent per-stage choices. The energy predictor consumes the 16 genotype variables plus parameter count, FLOPs, and total activations (the 19-dimensional x_{arch} of Section 4.1).

G Sensitivity and Robustness Analyses

PCA routing-threshold sweep Table 5 reports the leave-one-out routing decision of every GPU as the threshold multiplier m in $\tau = m \times \tilde{x}_E$ varies. Decisions are identical throughout $m \in [1.25, 2.5]$. At the conservative extreme ($m = 1.0$) the heavily factory-overclocked RTX 3090 is additionally flagged as an outlier; at $m \geq 3.0$ the RTX 5090 becomes an inlier while the A100 remains an outlier at every tested multiplier.

Table 5: Routing decisions (O = outlier $\rightarrow$ Transformer, I = inlier $\rightarrow$ CatBoost) under threshold multipliers m. The deployed setting is $m = 2$.

Held-out GPU	E_{recon}	m=1.0	m=1.25	m=1.5	m=2.0	m=2.5	m=3.0	m=4.0
A100 40GB	0.407	O	O	O	O	O	O	O
RTX 5090	0.204	O	O	O	O	O	I	I
RTX 3090	0.085	O	I	I	I	I	I	I
L40S	0.081	I	I	I	I	I	I	I
RTX A6000	0.068	I	I	I	I	I	I	I
RTX A5000	0.061	I	I	I	I	I	I	I
RTX 4090	0.056	I	I	I	I	I	I	I

Composite-score hyperparameters On the 100-architecture / 12-epoch ground truth, the composite's Spearman correlation is flat over the shift constant, $\rho \in [0.862, 0.865]$ for $K \in \{2, 3, 4, 6, 8, 12\}$, with the worst true accuracy inside the composite's top-20 selection identical at every K. Across 200 random weightings of the three proxies drawn log-uniformly from $[0.25, 2.5]^3$, ρ stays within $[0.81, 0.89]$ (median 0.86) and the top-20 selection is invariant in every setting. The aggregation therefore contains no tuned hyperparameter to overfit—in contrast to learned multi-proxy estimators.

Validation at full training length The seven EENAS architectures trained with the complete 300-epoch recipe reach $[72.90, 74.62, 75.22, 76.28, 78.02, 79.32, 80.37]\%$ Top-1, with composite scores $[3.5784,\ 3.8303, 3.9254, 4.1311, 4.3061, 4.6164, 4.9281]$. The proxy ordering is preserved without a single inversion: Spearman $\rho = 1.000$, Kendall $\tau = 1.000$ (Pearson $r = 0.99$). While $n = 7$ limits statistical power, this confirms that the 12-epoch validation protocol transfers to the final training setting.

H Absolute Hardware Bounding under Prediction Uncertainty

Guard band A point prediction with $\leq 10\%$ MAPE is not a worst-case certificate. For a hard cap C_{max}, retaining only candidates with $\hat{E}(a, g) \leq 0.9\, C_{max}$ guarantees $E \leq C_{max}$ whenever the individual under-prediction does not exceed 10% of the true value. Our intended deployment workflow is: (i) search tens of thousands of candidates with the zero-shot predictor, applying the guard band when the cap is strict; (ii) retain a small Pareto set near the operating budget; (iii) physically measure only these finalists on the target before deployment, which certifies compliance.

Cost of miscalibrated bounding Both predictors preserve candidate ranking almost perfectly (Spearman ≥ 0.98, Table 1), so an unconstrained search is largely insensitive to routing; the router's

contribution is calibration, which budget-constrained selection consumes. In a fixed-candidate-pool experiment on the held-out A100—where CatBoost-only reaches 41.59% MAPE—selecting the best-scoring architecture under an energy budget with CatBoost-only falsely excludes $\approx 42\%$ of truly feasible candidates and utilises only $\approx 68\%$ of the budget, forfeiting 0.37 composite-score points relative to a measured-energy oracle; routed selection reduces this to $\approx 1.5\%$ false exclusions, $\approx 93\%$ budget utilisation, and 0.075 score regret (averaged over nine budget levels).

Comparison to few-shot calibration Methods such as HELP and PlatformX assume few-shot on-device measurements; the axis EENAS removes is target-device access. Simulating the strongest cheap calibrator—a scalar log-offset fitted on k architectures measured on the target—shows that on the A100, a 10-measurement calibration does not improve on routed zero-shot prediction (8.3% vs. 8.0% MAPE in our held-out simulation): the routed predictor's residual OOD error is not a correctable global bias, i.e., zero-shot already sits at the few-shot-calibration frontier there. On in-distribution targets, ≈ 10 measurements buy a few MAPE points and can be applied as an optional calibration mode within the same pipeline when device access exists.

I Measurement-Regime Ablation: Why Cheap Energy Measurement Cannot Guide the Search

A natural question is whether one could simply measure energy approximately during the search—e.g., while executing the accuracy proxy's forward pass—instead of learning a predictor. We measured four MobileNetV4 models on an RTX 5090 under five regimes of increasing rigour, all at fp16 so that numerical precision is constant and each step isolates one factor:

1. Single forward pass: batch size 1, one forward pass in plain PyTorch `eval()`, with power recorded during just that pass (reported for the first invocation and for warmed repeats).
2. Batch-1 steady state: batch size 1 in plain PyTorch, measured over a ≥ 10 s / ≥ 12-batch steady-state window after ≥ 5 s warm-up (the protocol's window rule).
3. PyTorch batch sweep: regime 2 repeated across a batch-size sweep, reporting the energy-optimal point.
4. TensorRT, batch 1: batch size 1 compiled with the deployment stack, same steady-state window.
5. Full protocol: TensorRT with a batch-size sweep to the energy-minimal operating point under stability-based stopping (the protocol of Section 4.1).

Table 6: Per-image energy (mJ) of MobileNetV4 models under the five measurement regimes (RTX 5090, all fp16).

Measurement regime	Conv-S	Conv-M	Conv-M	Conv-L	Error vs. protocol
Input resolution	224	224	256	256	—
1a. Single pass, first invocation	114,532	241,790	235,909	227,371	$\sim 10^4\times$
1b. Single pass, warmed (mean of 4)	134	232	222	299	12–41×
2. Batch 1, steady state, PyTorch	188	329	323	451	19–57×
3. Best over batch sweep, PyTorch	10.3	30.3	36.9	61.3	2.6–3.2×
4. Batch 1, steady state, TensorRT	26.3	49.9	51.4	82.5	3.5–8×
5. Full protocol (optimal batch size)	**3.27**	**10.38**	**13.29**	**23.67**	1.0×

Three observations. (i) The literal "record power during the proxy's forward pass" measurement is unusable: a 3–6 ms pass is far below the temporal resolution of GPU power telemetry ($\sim$10–50 Hz with internal averaging; see [40]), yielding values 12–41× too high once warm, $\sim 10^4\times$ too high on first invocation, and up to $\sim 30\%$ spread between identical repeats. (ii) Cheaper steady-state variants remain miscalibrated relative to deployment-representative values by 2.6–57× depending on operating point and stack, and can invert rankings between nearby configurations: at batch size 1, the 256-px Conv-M measures as *cheaper* than the 224-px variant, whereas the protocol shows it costs $+28\%$. (iii) The full protocol takes $\approx$5–20 minutes per architecture *on the target device*; across a search's candidate evaluations this is the 44–178 target-GPU-day figure of Appendix J, versus milliseconds per candidate—without target access—for the zero-shot predictor. Incidentally,

this rerun independently reproduced the protocol's published Fig. 1 energy values within $\pm 2.4\%$, confirming measurement stability across separate campaigns.

J Measurement-Campaign Cost Accounting

The one-time 2,000-architecture $\times$ 7-GPU campaign consumed a conservative lower bound of $\approx$123 kWh (per-GPU campaigns 13–26 kWh), reconstructed from logged GPU power over the mandatory 15-second active measurement windows across all tested batch sizes: 14,000 measured architecture–GPU pairs at ≈ 8.8 Wh per pair (≈ 61.5 Wh per architecture across all seven GPUs). This excludes host power, TensorRT engine compilation, idle intervals, and orchestration overheads.

The relevant comparison is the repeated cost the surrogate replaces. The nominal search configuration evaluates $200 \times 64 = 12{,}800$ candidates before caching; applying the measurement protocol to these candidates on one target GPU would consume $\approx 12{,}800 \times 8.8$ Wh ≈ 112.5 kWh *per search*, so the dataset cost is amortised within approximately two searches—after which it is reusable for further searches and previously unseen NVIDIA targets within the represented architecture space and inference stack. In wall-clock terms, at 5–20 minutes per properly measured architecture, direct profiling of one search's candidates would require $\approx$44–178 target-GPU-days, versus under 30 minutes for the offline surrogate-based search.

K Search vs. Search-Space Contribution

To separate the contribution of the search from the quality of the search space itself, we compare the Pareto front of the raw LHS sampling pool (2,000 architectures with measured RTX 5090 energy—the space alone, with no search) against the searched final candidates. The search dominates at every matched energy budget despite the pool containing $31\times$ more points: the best proxy score at budgets of $\leq 5/7.5/10/20$ mJ improves by $+1.4/+1.6/+1.7/+2.6$ (in Zen-score units; pool standard deviation 4.4), and the search concentrates 16% of its candidates below 5 mJ where only 0.5% of LHS samples fall. The reported gains are therefore attributable to the search and predictors, not to the search space alone.

L Baseline Comparability Scope

Fig. 1 evaluates publicly released pretrained checkpoints, which reflect per-model training pipelines: distillation-based SSLD pretraining for HGNet-V2, and substantially larger, per-variant-tuned schedules for MobileNetV4. Fig. 4 instead compares architectural quality under one identical 300-epoch recipe at 224×224. The two views are not numerically interchangeable; recipe effects alone can move a fixed architecture by more than the gaps in question (e.g., ResNet-50 from 75.3% to 80.4%; [36]).

Table 7: Published ImageNet-1K-only MobileNetV4 training configurations vs. the unified recipe used in Fig. 4.

Training setting	MNv4 published (Conv-S / Conv-M)	Unified recipe (all models)
Epochs	9,600 / 500	300
Input resolution	224 / 256	224, fixed for all models
Batch size	4,096 / 4,096–16,384	1,024
Peak learning rate	tuned per variant: 0.002 / 0.004	10^{-3}, identical
Weight decay	tuned per variant: 0.01 / 0.1	0.05, identical
Dropout	tuned per variant: 0.3 / 0.2	identical for all
Stochastic depth	tuned per variant: 0–0.075	0.1, identical
RandAugment magnitude	tuned per variant: 9 / 15	identical for all
Mixup / CutMix	disabled for Conv-S/M	applied to all models

Such training investments are architecture-agnostic: applied to EENAS architectures, they would shift all operating points upward together, so published-checkpoint deltas measure training compute and per-model tuning rather than architectural merit—the quantity Fig. 4 isolates.

M Code Repository

Our implementation is available at:

`https://github.com/JimZeyuYang/energy-efficiency-aware-nas`

The repository contains the search space and model builder, the energy-measurement pipeline, the predictor training and routing code, the zero-cost proxies and composite score, the NSGA-II search, the ImageNet training recipe, and the genotypes of the seven EENAS models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and introduction accurately reflect our work in an unbiased way.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the Limitations paragraph in Section 6.2.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: Our paper is focused on modeling and empirical results, not theoretical.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We believe that we have included all information needed for an experienced researcher to reproduce our results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Please refer to the GitHub Repository link in the appendix.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We believe that we have included all training and test details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because each ImageNet model was trained once under the 300-epoch recipe due to its computational cost, and the energy-prediction results are aggregate metrics (MAPE and Spearman) over 400 held-out architectures per GPU rather than repeated runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have ensured that we have included a description of the GPUs we have used in this work.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have checked against the code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We believe we have made the impact of the work clear in our paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not believe this question is relevant to our submission.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We believe that all owners of assets are correctly acknowledged.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New asserts introduced in the paper are all clearly documented, with source code in the GitHub repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not believe this question is relevant to our work

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not believe that this question is relevant to our work

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We have only used an LLM to improve grammar.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.