AAAE: APPLICATION ACCURACY-AWARE EVALUATION FRAMEWORK FOR APPROXIMATE COMPUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximate computing offers an effective approach to reducing the hardware cost of neural inference, but efficiently characterizing model-specific error tolerance across large approximate multiplier design spaces remains challenging. We present AAAE, an application accuracy-aware evaluation framework that guides approximate multiplier design under explicit accuracy constraints. AAAE combines multi-scale and model-conditioned error features, augmented with lightweight inference responses for Transformers, to predict the accuracy degradation of previously unseen multiplier designs. In parallel, a graph neural network estimates hardware costs from multiplier structures. These predictions, together with empirically calibrated accuracy bounds, guide design optimization, with selected designs verified through inference and high-fidelity hardware evaluation. Experiments on three vision Transformers and two CNNs demonstrate Top-1 accuracy-drop prediction mean absolute errors of 0.43–0.59 percentage points. The hardware surrogate achieves a power–delay product prediction mean relative error of 0.241%. In the reported timing setting, surrogate evaluation provides a $35.1\times$ per-design speedup over $1{,}000$-image accuracy evaluation plus high-fidelity hardware evaluation. Under a two-percentage-point accuracy-loss budget, the calibrated bound A1 achieves an empirical false-discovery proportion of at most 1% across all five models. Our implementation is publicly available at `https://anonymous.4open.science/r/AAAE-7CB0/`.

1 INTRODUCTION

Deep neural networks (DNNs) are increasingly deployed in battery-powered edge devices, embedded vision systems, and domain-specific AI accelerators, where practical deployment is constrained by strict power, performance, and area (PPA) requirements. Since convolution and matrix multiplication dominate the computation cost of DNN inference, optimizing multiplication hardware is essential for improving system efficiency. Approximate multiplication trades arithmetic precision for reduced hardware overhead while maintaining acceptable application-level accuracy (Armeniakos et al., 2022; Hu et al., 2024; Wu et al., 2024; Yang et al., 2018). Many inference applications can tolerate limited accuracy degradation under application-level accuracy constraints, which specify the maximum allowable accuracy loss relative to a reference model, with thresholds of 0.5%–1% and 0.5%–2% explored in previous studies (Spantidi et al., 2021; Panteleaki et al., 2025). However, the achievable accuracy–efficiency trade-off strongly depends on approximation error characteristics, target neural networks, workloads, and cross-layer interactions (Armeniakos et al., 2022; Leon et al., 2025; Wu et al., 2024).

The major challenge in approximate multiplier design is no longer the construction of low-cost arithmetic units, but the efficient identification of designs that simultaneously satisfy application-level accuracy constraints, such as a 5% accuracy-loss budget for image classification (Ponzina et al., 2022) or a 2% relative perplexity-increase budget for language modeling (Xiao et al., 2023), while providing meaningful hardware benefits. Reliable evaluation requires multiple levels of analysis, including circuit-level error characterization, model-level emulation of approximate multiplication within neural network execution, inference accuracy validation, and high-fidelity PPA estimation. However, each stage introduces significant computational overhead. Model-level evaluation requires repeated inference for every multiplier design, while accurate hardware evaluation depends on costly synthesis and timing analysis of digital IC design flow. Meanwhile, the design space of

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.

approximate multipliers is extremely large, involving diverse architectural choices such as bit-width selection, truncation schemes, compressor structures, partial-product reduction, sign processing, and error compensation. Recent automated approximate circuit generation frameworks and design methodologies further accelerate design generation, making exhaustive evaluation increasingly impractical (Zhou et al., 2026a; Mrazek et al., 2019a; Meng et al., 2026).

Although existing studies have improved application-level evaluation in approximate computing (Mrazek et al., 2019a; Jain et al., 2022; Danopoulos et al., 2025), reliably and efficiently estimating the accuracy degradation caused by approximate multipliers remains challenging. Conventional circuit-level error metrics provide inexpensive summaries of arithmetic quality (Yang et al., 2018; Armeniakos et al., 2022; Hu et al., 2024; Wu et al., 2024), but they cannot capture how structured multiplier errors interact with model-specific computation to affect inference accuracy. Learning-based approaches predict accuracy degradation from selected statistical error features (Mo et al., 2024), while the architectural mean error (AME) combines complete error matrices with approximate propagation models to estimate convolutional neural network (CNN) accuracy degradation (Liu et al., 2024). Meanwhile, emulation and retraining techniques improve the practicality of approximate inference and model robustness against fixed approximate operators (Danopoulos et al., 2023; Gong et al., 2023; Meng et al., 2025), and multi-fidelity optimization reduces hardware evaluation cost (Sakhuja et al., 2024). However, statistical features inevitably discard part of the detailed error structure, whereas analytical propagation models rely on simplified assumptions about how approximation errors propagate through network computation. Moreover, existing studies have predominantly focused on CNNs, with relatively limited attention to Transformers, where the interactions among self-attention, residual connections, and normalization introduce more complex error propagation. Consequently, reliably and efficiently evaluating end-to-end accuracy degradation and characterizing model-specific error tolerance remain challenging, particularly for Transformers and across large design spaces. A detailed discussion of related work is provided in Appendix A.

In this work, we introduce AAAE, an application accuracy-aware evaluation framework that supports the design and optimization of approximate computing under explicit application accuracy constraints. The main contributions of this work are threefold: (1) We propose an application-aware framework for evaluating the error tolerance of approximate computing, integrating approximate multiplier generation, accuracy prediction, and hardware efficiency estimation into a unified exploration flow. By incorporating calibrated accuracy margins, it efficiently identifies multiplier designs that satisfy application accuracy constraints while achieving improved PPA efficiency. (2) We develop target-model-aware predictors with family-specific inputs and heads. Both architecture families combine complete multiplier error behavior with the target model's input-value distribution; the Transformer path additionally uses a fixed, lightweight labeled response set. The predictors estimate the Top-1 accuracy degradation of previously unseen multiplier designs without exhaustive inference evaluation. (3) We build an efficient approximate multiplier generation and characterization toolkit that supports large-scale design generation, comprehensive error analysis, and surrogate predictor training, enabling scalable approximate-multiplier design-space exploration for application-aware evaluation and optimization.

2 METHODOLOGY

Figure 1 summarizes the pipeline: multiplier representations (Section 2.1), model-conditioned accuracy prediction (Section 2.2), and structure-based hardware-cost estimation (Section 2.3) feed the accuracy-constrained design optimization of Section 2.4.

The application-level objective is formalized as follows. Let $\mathcal{A}$ denote the approximate-multiplier design space, f_m the baseline model using exact multiplication, and $f_{m,a}$ the same model using design $a \in \mathcal{A}$ under a fixed deployment protocol. The Top-1 accuracy drop is

$$y_{m,a} = 100 \times \left[\mathrm{Acc}(f_m; \mathcal{D}_{\mathrm{eval}}) - \mathrm{Acc}(f_{m,a}; \mathcal{D}_{\mathrm{eval}})\right], \tag{1}$$

where Acc denotes Top-1 accuracy on the fixed 1,000-image evaluation set $\mathcal{D}_{\mathrm{eval}}$ (Section 3.1) with $\mathrm{Acc} \in [0, 1]$, $y_{m,a}$ is expressed in percent, and $y_{m,a} < 0$ records an accuracy improvement. Given an allowable budget τ, the design objective minimizes hardware cost subject to the constraint

$$a^* = \arg\min_{a \in \mathcal{A}} \mathrm{PDP}(a) \quad \text{subject to} \quad y_{m,a} \le \tau, \tag{2}$$

with $\mathrm{PDP}(a) = \mathrm{Power}(a) \times \mathrm{Delay}(a)$.

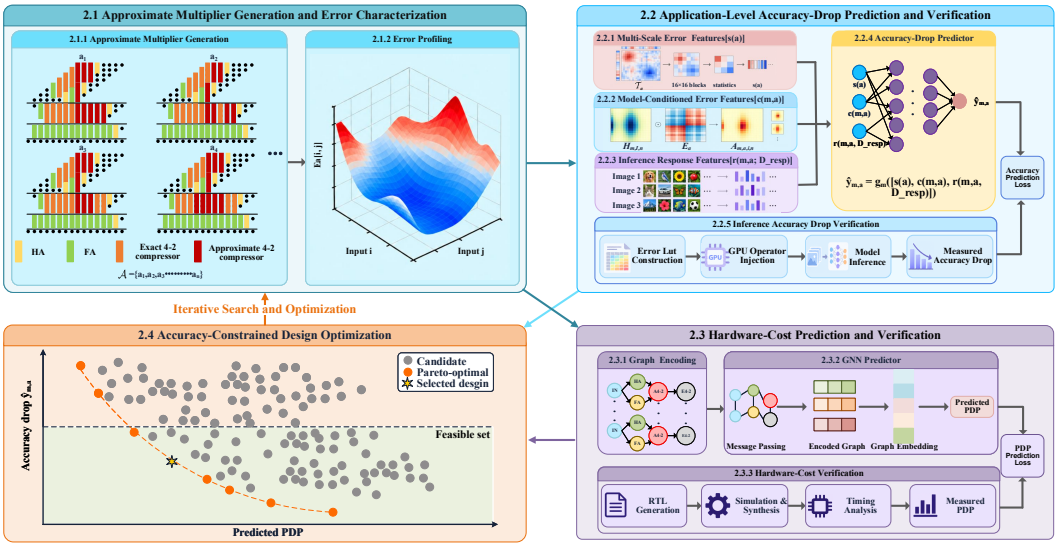
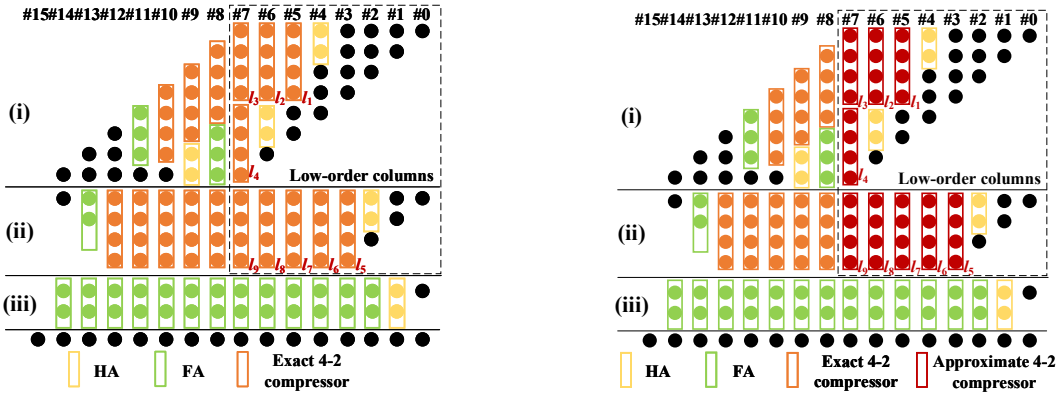


Figure 1: Overview of AAAE. The accuracy block denotes the unified predictor of equation 4; the Transformer is the main instantiation, and CNNs use a response-free extension. The calibrated accuracy bound and the predicted hardware cost drive the constrained design optimization, and selected designs are verified by full-model inference and high-fidelity PPA analysis.

2.1 Multiplier Generation and Error Characterization

2.1.1 Approximate Multiplier Generation



(a) Exact multiplier

(b) Approximate multiplier

Figure 2: Compressor-based reduction topology of a generated 8-bit multiplier. (a) Exact multiplier. (b) Approximate multiplier, where the nine configurable low-order positions $l_1, \ldots, l_9$ hold the selected approximate compressor cells while the remaining eight high-order positions keep exact compressors; in both panels the partial-product reduction proceeds through three stages, and HA and FA denote half and full adders.

AAAE generates 8-bit approximate multipliers within a fixed compressor-based reduction topology. A design is represented by $\theta_a = (l_1, \ldots, l_9)$, which selects one of eight approximate compressor cells drawn from prior multiplier designs (Wu et al., 2024) at each of nine configurable low-order positions, while the remaining eight high-order positions keep exact compressors (Figure 2); designs therefore differ only in local compressor choices. The generator derives the corresponding register-transfer-level (RTL) implementation and structural compressor graph G_a (used by the hardware-cost

predictor of Section 2.3); configurations with identical arithmetic behavior are grouped as described in Section 3.1.

2.1.2 GPU-ACCELERATED COMPUTATION OF TRUTH TABLES AND ERROR FEATURES

To characterize each design completely, AAAE exhaustively evaluates all 256×256 input-value pairs of the 8-bit input domain as batched tensor operations, where each approximate compressor is represented by its local truth table, enabling parallel construction of complete multiplication tables without per-design RTL simulation (Danopoulos et al., 2023; Gong et al., 2023). The resulting unsigned table U_a is converted to the signed S8×S8 domain by applying the operand signs to the unsigned magnitudes, yielding the table T_a, and the error surface of design a is

$$E_a[i, j] = T_a[i, j] - (i - 128)(j - 128), \tag{3}$$

indexed by the signed input codes $i, j \in \{0, \ldots, 255\}$. The complete surface E_a preserves both the magnitude and the input-dependent structure of approximation errors and is the arithmetic-error representation used by the application-conditioned modeling in Section 2.2.

2.2 APPLICATION-LEVEL ACCURACY-DROP PREDICTION AND VERIFICATION

Circuit-level statistics describe a multiplier's average error but omit how the target model's computation interacts with and propagates that error. The accuracy branch of Figure 1 predicts the proportion-scale Top-1 drop from three complementary sources of information: multiplier-specific structured error features $s(a)$, which answer what errors a design can produce; model-conditioned error features $c(m, a)$, which answer where those errors appear in the target model and how they affect its decisions; and lightweight inference-response features $r(m, a, \mathcal{D}_{\text{resp}})$, which capture the residual functional effect observed under a small budget of real approximate inference,

$$\widehat{y}_{m,a} = g_m\big([s(a),\, c(m, a),\, r(m, a, \mathcal{D}_{\text{resp}})]\big). \tag{4}$$

2.2.1 MULTI-SCALE ERROR FEATURES

Starting from the complete error surface E_a obtained in Section 2.1.2, AAAE constructs a four-channel error representation to capture signed bias, error magnitude, and positive–negative asymmetry,

$$\boldsymbol{\mathcal{T}}_a = \big(E_a, |E_a|, \max(E_a, 0), \max(-E_a, 0)\big). \tag{5}$$

For each channel, we extract 256 row means, 256 column means, and 256 spatial statistics from a 16×16 grid to preserve error characteristics at different scales. The resulting 3,072 projection features are combined with 12 global error statistics and 19 generator descriptors. After signed-log transformation, these features form the 3,103-dimensional multi-scale error representation $s(a)$, which serves as the multiplier-specific input to the accuracy-drop predictor. Appendix B.1 lists the exact statistics, dimensions, and normalization rules.

2.2.2 MODEL-CONDITIONED ERROR FEATURES

The same approximate multiplier can affect different neural models differently, as multiplier errors interact with model-specific input-value distributions and propagate through different computational paths. To characterize these interactions, AAAE combines the complete multiplier error surface with the target model's input-value distribution and decision sensitivity. Here i indexes profile images and n denotes a profiled computation site. For Transformers, we profile 48 weight-bearing Linear sites across 12 blocks (qkv, proj, $\mathrm{fc1}$, and $\mathrm{fc2}$ in each block), excluding attention qk, attention av, and downsampling operations.

For each image i and site n, we accumulate the product-scale mass $S_{m,i,n}(u, v)$ associated with each ordered input-value pair (u, v). After removing semantic-zero multiplications and normalizing the remaining mass, we obtain the ordered joint input distribution

$$H_{m,i,n}(u, v) = \frac{S_{m,i,n}(u, v)}{\sum_{u',v'} S_{m,i,n}(u', v')}. \tag{6}$$

This distribution characterizes how the target model uses different input-value pairs at each computation site. Unlike separate activation and weight distributions, it preserves input ordering and captures asymmetric multiplier error behavior.

Contracting $H_{m,i,n}$ with the error surface E_a gives the expected residual amplitude at site n for image i,

$$A_{m,a,i,n} = \langle H_{m,i,n}, E_a \rangle_F. \tag{7}$$

Because identical arithmetic residuals affect predictions differently depending on where they occur, we measure the local decision sensitivity $\rho_{m,n,i}$ by injecting a small residual $\pm\epsilon$ at the output of site n and observing the induced change in the clean winner–runner-up logit margin M_i. The resulting design-specific propagated effect is

$$\xi_{m,a,i,n} = A_{m,a,i,n}\rho_{m,n,i}. \tag{8}$$

We aggregate these propagated contributions across computation roles, network depth, image groups, clean-margin ranges, and decision-boundary-related summaries to construct the Transformer-specific model-conditioned error representation $c_{\mathrm{T}}(m, a)$, denoted by $c(m, a)$ in the unified prediction framework. Because H, ρ, and M are precomputed once per target model, evaluating a new design requires only contractions and aggregation with its error surface E_a, with no additional model inference. Detailed aggregation rules are provided in Appendix B.2.

2.2.3 Inference Response Features Evaluation

Static encoding and clean-model conditioning both approximate the effect of full execution, so AAAE adds a low-cost functional probe: each Transformer design is evaluated on a fixed labeled response set S_{50} of 50 images, selected using training designs only; this size follows from the response-budget study of Section 3.3. The probe complements rather than replaces full evaluation. For $k \in \{1, 5\}$, the correctness transition

$$F^{(k)}_{a,j} = \mathbf{1}\{y^{\mathrm{cls}}_j \in \mathrm{Top}\, k(z_j)\} - \mathbf{1}\{y^{\mathrm{cls}}_j \in \mathrm{Top}\, k(\widetilde{z}_{a,j})\} \in \{-1, 0, 1\} \tag{9}$$

gives a discrete functional change, recording a lost ($F = 1$) or recovered ($F = -1$) prediction between exact logits z_j and approximate-model logits $\widetilde{z}_{a,j}$ under design a; here y^{cls}_j is the ground-truth class label of image j, $\mathbf{1}\{\cdot\}$ denotes the indicator function, and $\mathrm{Top}\, k(\cdot)$ returns the k highest-scoring classes. The Top-1 and Top-5 margin changes provide a finer-grained continuous signal. We encode the Top-1/Top-5 correctness transitions and margin changes using aggregate statistics and ordered per-image responses, and additionally include residuals relative to a target-model-specific static prior $\pi^{(k)}_{m,a}$, emphasizing functional information not captured by $s(a)$; the resulting response representation is denoted $r(m, a, \mathcal{D}_{\mathrm{resp}})$.

2.2.4 Feature Fusion and Accuracy-Drop Prediction

In the fusion block of Figure 1, AAAE instantiates equation 4 with an independent Light Gradient Boosting Machine (LightGBM) regressor (Ke et al., 2017) g_m for each target model, which maps the three representations to the full-set Top-1 drop. For a newly generated design it then runs approximate-model inference on the 50 response images and returns $\widehat{y}_{m,a}$ for the accuracy-constrained optimization of Section 2.4. Fitting and refitting details are given in Appendix B.3.

CNN extension. CNNs use the same principle without response features, with a lightweight predictor over factorized input distributions and layer sensitivities (Appendix B.4).

2.2.5 Inference Accuracy Drop Verification

Full-model measurement provides Top-1 labels $y_{m,a}$ for predictor training, validation, calibration, testing, and selected-design verification. AAAE uses the signed S8×S8 error surface E_a from Section 2.1.2 as a deterministic 256×256 INT32 residual LUT indexed by raw codes $q_x + 128$ and $q_w + 128$. CUDA custom operators add scale-adjusted residual sums to the exact Q/DQ MatMul or Conv outputs, masking residuals at the quantization zero point.

We inject one candidate LUT into all 48 weight-bearing Linear MatMuls per Transformer, 53 Conv nodes in ResNet-50 (He et al., 2016), or 62 Conv nodes in MobileNetV3-Large (Howard et al., 2019); other operations retain exact quantized-model behavior. We run the exact quantized baseline and injected model on the same 1,000 labeled images under identical preprocessing, input order, batch size, and session profile. Equation equation 1 defines their paired Top-1 difference in percent; we multiply stored proportion-scale drops by 100. The 50-image Transformer response in Section 2.2.3 uses the same injection backend.

2.3 HARDWARE-COST PREDICTION AND VERIFICATION

To avoid the computational overhead of repeated hardware synthesis and PPA evaluation in large-scale design-space exploration, we employ a graph-based surrogate for hardware-cost estimation. The hardware branch of Figure 1 represents each approximate multiplier as a graph of its compressor configuration and structural dependencies, and a graph neural network (GNN) predicts power, delay, and area from this representation.

2.3.1 GRAPH ENCODING OF APPROXIMATE MULTIPLIER

Each candidate configuration θ_a is encoded as a graph $G_a = (V, E)$, where the nodes V represent half adders (HAs), full adders (FAs), and exact and approximate 4-2 compressors, while the edges E describe signal connections between arithmetic units. Nodes are organized by compression stage, and each node feature combines the input-combination probabilities propagated through the compressor truth table with the corresponding arithmetic error.

2.3.2 GRAPH NEURAL NETWORK FOR PDP PREDICTION

A stage-wise GraphSAGE network performs message passing to aggregate information from neighboring arithmetic units and update node embeddings, capturing the structural and functional characteristics of different compressor configurations. The embeddings of approximate compressor nodes and final-stage nodes are then concatenated to form a global graph representation. Finally, a multilayer perceptron (MLP) predicts power, delay, and area, from which the PDP of each candidate multiplier is calculated.

2.3.3 HARDWARE-COST VERIFICATION

For hardware-cost verification, selected multiplier configurations are evaluated through an EDA flow using the RTL implementations generated in Section 2.1. Following logic synthesis, SDF-annotated gate-level simulation and static timing analysis are performed to estimate power and critical-path delay. Area is obtained from synthesis, and the post-synthesis PDP is compared with the surrogate prediction.

2.4 ACCURACY-CONSTRAINED DESIGN OPTIMIZATION

For a target model m, AAAE explores the design space $\mathcal{A}$ under a Top-1 accuracy-drop budget τ using Bayesian optimization (Snoek et al., 2012) and NSGA-II (Deb et al., 2002), guided by the learned accuracy and hardware surrogates. For fixed-pool evaluation, we consider $\mathcal{C} \subseteq \mathcal{A}$. The accuracy predictor is selected using Train and Validation and refitted on their union, while a separate Calibration set provides a pool-weighted, one-sided empirical correction $q^{(\mathcal{C})}_{m,\alpha}$. The resulting admission bound is

$$U^{(\alpha)}_{m,a} = 100\widehat{y}_{m,a} + q^{(\mathcal{C})}_{m,\alpha}. \tag{10}$$

This empirical bound does not provide a finite-sample coverage guarantee.

Designs satisfying $U^{(\alpha)}_{m,a} \leq \tau$ are admitted and ordered by predicted PDP. At most K designs undergo full accuracy evaluation, and the feasible design with the lowest predicted PDP is selected. Empty admission constitutes abstention, while the absence of a feasible evaluated design constitutes failed selection. The selected design receives a same-image inference rerun and high-fidelity PPA verification, with any verification failures reported without changing the selection. This fixed-pool evaluation uses a fixed accuracy-query budget and does not construct an iterative Pareto front. Detailed calibration and selection procedures are provided in Appendices C and D.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP AND EVALUATION PROTOCOL

We evaluate five target models, Swin-Tiny (Liu et al., 2021), ViT-Base (Dosovitskiy et al., 2021), DeiT-Base (Touvron et al., 2021), ResNet-50 (He et al., 2016), and MobileNetV3-Large (Howard

et al., 2019), each under the fixed operator scope of Section 2.2.5. Candidate designs are drawn from a 20,000-design pool that supplies 14,000 Train, 3,000 Validation, and 3,000 Test configurations, and a separately generated 3,000-design pool supplies the Calibration candidates; a structural-uniqueness audit found no duplicate structures, so no design appears in more than one split.

All experiments share one measurement protocol: the paired Top-1 drop $y_{m,a}$ is measured on $\mathcal{D}_{\text{eval}}$ against the exact quantized baseline of equation 1. Profiling and evaluation images use separate roles, whose identities, the post-selection audit set $\mathcal{D}_{\text{audit}}$, and the reporting conventions are recorded in Appendix B.5. Hardware results report the PDP of a multiplier in femtojoules (fJ) under a single specified circuit flow, and the split manifests are given in Appendix B.6. All experiments are performed on a workstation with an Intel Xeon Gold 6330 CPU (2.00 GHz) and an NVIDIA A100 GPU (40 GB).

3.2 Accuracy-Drop Prediction on Approximate Multipliers

We predict the Top-1 drop $y_{m,a}$ of unseen multiplier designs on a structural hold-out Test set of 3,000 candidates per target model. All methods share one candidate split, label set, and evaluation set. Table 1 reports MAE / Spearman ρ, plus AAAE's absolute P95 and maximum error. We refit the cited Statistical Error Features and AME constructions on our candidate pool under the same protocol, so every number comes from our approximate multipliers.

Table 1: Top-1 drop prediction on the structural hold-out Test set (3,000 candidates per model). Each row is a method and each column a target model; for the first four rows each cell reports MAE / Spearman ρ, while the last two rows report AAAE's absolute P95 and maximum absolute error. All errors are absolute Top-1 accuracy differences in percentage points. Baseline rows follow the cited constructions, refit on our candidate pool. N/A marks methods outside that model family; metric definitions are given in Appendix B.7.

Method	Swin-Tiny	ViT-Base	DeiT-Base	ResNet-50	MobileNetV3-Large
Statistical Error Features (Mo et al., 2024)	2.858 / 0.911	5.515 / 0.948	0.805 / 0.976	0.662 / 0.975	1.030 / 0.992
AME (Liu et al., 2024)	N/A	N/A	N/A	2.501 / 0.893	6.105 / 0.967
Response-only	0.811 / 0.987	0.957 / 0.994	0.669 / 0.975	N/A	N/A
AAAE (Ours)	**0.542 / 0.994**	**0.574 / 0.997**	**0.432 / 0.982**	**0.586 / 0.985**	**0.546 / 0.997**
AAAE P95 AE	1.546	1.783	1.316	1.397	1.569
AAAE max AE	4.428	5.803	4.717	3.779	4.938

The Statistical Error Features baseline (Mo et al., 2024) trains on the target model's Train labels. AME (Liu et al., 2024) combines the error LUT and the target CNN's application matrix into 11 features and fits a quadratic-feature Ridge model, so it covers the CNNs only. The Response-only regressor shares AAAE's 50 response images and response features, so it covers the Transformers only; Section 3.3 isolates its contribution. AAAE uses $s + c + r$ for Transformers and $s + c$ for CNNs. We select all hyperparameters on Validation, evaluate once on Test, and report Transformer and CNN groups separately. ViT-Base and DeiT-Base form sealed evaluations, Swin-Tiny a post-inspection promotion result, and ResNet-50 and MobileNetV3-Large post-failure audits.

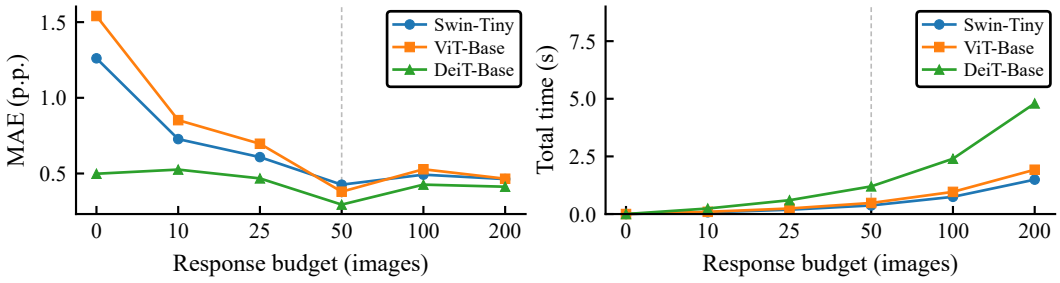
Appendix E illustrates post hoc candidate pairs that conventional metrics match closely yet whose measured drops diverge.

3.3 Feature Ablations and Response Budget Analysis

The ablation holds the candidate split and fitting protocol fixed while removing or replacing $s(a)$, $c(m, a)$, and the response features r. Table 2 reports MAE, Spearman ρ, and tail errors for the retrained Transformer and CNN variants. The response-budget experiment compares error with measured response-plus-predictor time per candidate. Figure 3 reports the comparison: prediction error drops sharply up to the 50-image operating point and then stays flat, while the response-plus-prediction cost grows linearly with the budget.

Table 2: Controlled feature ablation under a fixed retraining protocol (macro averages over three Transformers and two CNNs). Here s, c, and r denote the static, conditioned, and response features of Section 2.2. Each column reports MAE, Spearman ρ, absolute P95, and maximum absolute error; all errors are absolute Top-1 accuracy differences in percentage points. These retrained variants are not directly comparable to the sealed models of Table 1. Metric definitions are given in Appendix B.7.

Model	Inputs	MAE ↓	Spearman ρ ↑	Abs. P95 ↓	Max abs. err. ↓
Trans.	s-only	1.148	0.984	4.226	29.020
	$s + c$	1.094	0.985	3.967	23.518
	$s + c + r$ (Ours)	**0.516**	**0.991**	**1.548**	**4.983**
CNN	s-only	0.630	0.990	1.660	7.186
	$s + c$ (Ours)	**0.566**	**0.991**	**1.483**	**4.359**



(a) Prediction error

(b) Response + prediction cost

Figure 3: Response-budget study for the three Transformers. (a) Prediction MAE on the Test candidates, in percentage points, versus the number of response images, with the 50-image operating point marked; (b) total response-plus-prediction time per candidate versus the budget.

3.4 Calibrated Design Optimization under Accuracy Constraints

Table 3: Screening results at $\alpha = 0.10$ and $\tau = 2$ % for five target models. Each row is an admission rule and each cell reports coverage / FDP ↓ (%) / feasible recall ↑ (%); only A1/A2 use a nominal coverage level of 0.90. Definitions are given in Appendix B.7.

Rule	Swin-Tiny	ViT-Base	DeiT-Base	ResNet-50	MobileNetV3-Large
Point	0.499/10.9/81.0	0.486/16.3/87.4	0.474/3.1/99.2	0.794/0.2/84.3	0.502/16.0/96.0
Fixed-Margin	0.928/0.0/11.6	0.914/0.0/33.6	0.953/0.1/79.7	0.972/0.0/4.3	0.951/0.6/70.9
A1	0.887/0.0/24.6	0.892/0.9/40.8	0.827/1.0/92.6	0.965/0.0/22.5	0.975/0.6/63.3
A2	0.889/3.0/56.0	0.853/0.0/33.6	0.812/1.0/88.4	0.920/0.2/79.5	0.917/0.0/58.2

We compare four admission rules on the same fixed design pool: Point, Fixed-Margin, and the calibrated bounds A1 and A2. A1 applies the pool-weighted empirical correction $q^{(\bar{C})}_{m,\alpha}$ of equation 10 uniformly to every candidate, whereas A2 recomputes the same quantile separately inside each prediction bin (Mondrian), so its correction depends on the predicted-drop bin. Neither correction carries a finite-sample split-conformal guarantee, so all rules are evaluated empirically. Table 3 reports results at $\alpha = 0.10$ and $\tau = 2$ %, using coverage, false-discovery proportion (FDP), and feasible recall. The Point rule achieves coverage of 0.47–0.79 with FDP up to 16.3 %, whereas the calibrated bounds achieve coverage of 0.81–0.98 with FDP at most 3 %. A1 also achieves higher feasible recall than Fixed-Margin on four of five models. Complete results, including $\tau = 0.5$ % and empty-admission cases, are provided in Appendix C.

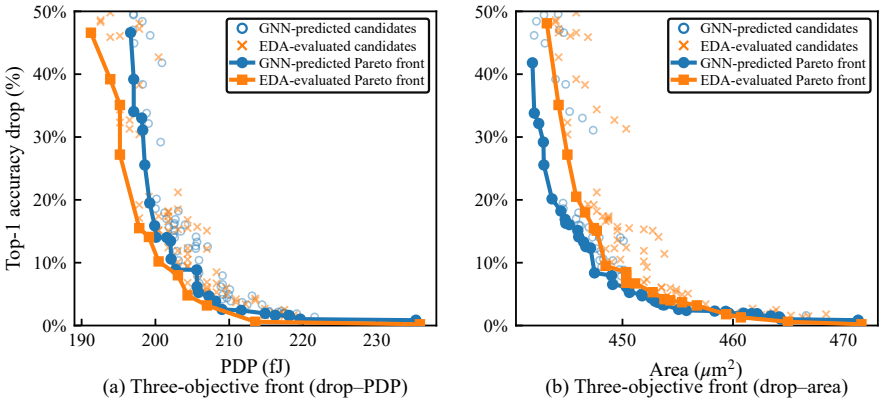
3.5 Hardware-Cost Prediction Accuracy

Table 4: Prediction accuracy of the GNN-based hardware-cost predictor (definitions: Appendix B.7).

Metric	MSE ($\times 10^{-4}$)	MRE (%)	R^2
Delay	0.0534	0.0012	0.9997
Area	0.6188	0.0560	0.9961
Power	2.0295	0.2409	0.9895
PDP	**2.0270**	**0.2410**	**0.9896**

Table 4 reports the prediction accuracy of power, delay, area, and PDP for the GNN-based hardware-cost predictor on an independent test set of 8-bit approximate multipliers, using mean squared error (MSE), mean relative error (MRE), and the coefficient of determination (R^2). The GNN achieves an MRE below 0.25% and an R^2 above 0.989 across all four metrics, demonstrating its accuracy in estimating multiplier hardware costs. One design takes about 0.24 s through the surrogate, against about 34 s for the synthesis, gate-level simulation, and static timing analysis of Section 2.3.3, a speedup of more than 140×.

3.6 Approximate Multiplier Design Optimization under Accuracy Constraints



(a) Three-objective front (drop–PDP) (b) Three-objective front (drop–area)

Figure 4: Predicted and EDA-verified Pareto fronts from accuracy-constrained Bayesian optimization, projected onto Top-1 accuracy drop versus (a) PDP and (b) area. Faded markers denote designs, and bold curves denote Pareto fronts; blue indicates GNN predictions, while orange indicates EDA measurements paired with true accuracy drops. Designs with accuracy drops above 50 % are omitted.

Bayesian optimization and NSGA-II search under the accuracy budget τ using learned accuracy and hardware surrogates. Figure 4 compares predicted and verified Bayesian fronts, with NSGA-II results in Appendix D. All front designs are evaluated on the fixed $\mathcal{D}_{\text{eval}}$, and selected designs receive high-fidelity PPA verification. At matched Top-1 drops of 0.9–47 %, the verified front shows a median PDP reduction of 2.7 fJ and a mean area difference of 2.1 μm^2; at $\tau = 2$ %, its lowest PDP and area are 213.5 fJ and 459.4 μm^2. The complete surrogate evaluation used by both optimization algorithms is 35.1× faster per candidate than 1,000-image accuracy evaluation plus high-fidelity PPA.

4 Conclusion

AAAE integrates model-aware accuracy prediction, GNN-based hardware-cost estimation, and calibrated optimization to enable efficient accuracy-constrained approximate computing design. Experiments on CNNs and Transformers demonstrate its effectiveness. Future work will extend model and multiplier coverage while reducing profiling costs and incorporating physical implementation.

AI use statement

In this work, we did not use generative AI tools for any of the tasks requiring disclosure, such as developing theoretical models, formulating or proving mathematical claims, proposing hypotheses,

designing experiments, implementing methods, cleaning or reformatting datasets, translating text, or interpreting results. We used generative AI tools for the following tasks with recommended disclosure: editing the manuscript for grammar and readability, reviewing our implementation code for bugs, and searching for relevant literature. All AI-assisted output was reviewed by the authors before being adopted: language edits were checked against the intended meaning, code-level suggestions were tested in our implementation, and every reference identified with AI assistance was verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Mohammad Saeed Ansari, Vojtech Mrazek, Bruce F. Cockburn, Lukas Sekanina, Zdenek Vasicek, and Jie Han. Improving the accuracy and hardware efficiency of neural networks using approximate multipliers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2), 2020. doi: 10.1109/TVLSI.2019.2940943.

Giorgos Armeniakos, Georgios Zervakis, Dimitrios Soudris, and Jörg Henkel. Hardware approximate techniques for deep neural network accelerators: A survey. *ACM Computing Surveys*, 55 (4):83:1–83:36, 2022. doi: 10.1145/3527156.

Dimitrios Danopoulos, Georgios Zervakis, Kostas Siozios, Dimitrios Soudris, and Jörg Henkel. AdaPT: Fast emulation of approximate DNN accelerators in PyTorch. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(6):2074–2078, 2023. doi: 10.1109/TCAD.2022.3212645.

Dimitrios Danopoulos, Georgios Zervakis, Dimitrios Soudris, and Jörg Henkel. TransAxx: Efficient transformers with approximate computing. *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, 2(4):288–301, 2025. doi: 10.1109/TCASAI.2025.3565685.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Jing Gong, Hassaan Saadat, Hasindu Gamaarachchi, Haris Javaid, Xiaobo Sharon Hu, and Sri Parameswaran. ApproxTrain: Fast simulation of approximate multipliers for DNN training and inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42 (11):3505–3518, 2023. doi: 10.1109/TCAD.2023.3253045.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019. doi: 10.1109/ICCV.2019.00140.

Xiaolu Hu, Ao Liu, Xinkuang Geng, Zizhong Wei, Kai Jiang, and Honglan Jiang. A configurable approximate multiplier for CNNs using partial product speculation. In *Design, Automation & Test in Europe Conference (DATE)*, 2024. doi: 10.23919/DATE58400.2024.10546701.

Paras Jain, Safeen Huda, Martin Maas, Joseph E. Gonzalez, Ion Stoica, and Azalia Mirhoseini. Learning to design accurate deep learning accelerators with inaccurate multipliers. In *Design, Automation & Test in Europe Conference (DATE)*, pp. 184–189, 2022. doi: 10.23919/DATE54114.2022.9774607.

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, pp. 3146–3154, 2017.

Vasileios Leon, Muhammad Abdullah Hanif, Giorgos Armeniakos, Xun Jiao, Muhammad Shafique, Kiamal Pekmestzi, and Dimitrios Soudris. Approximate computing survey, part II: Application-specific & architectural approximation techniques and applications. *ACM Computing Surveys*, 57 (7):1–36, 2025. doi: 10.1145/3711683.

Ao Liu, Jie Han, Qin Wang, Zhigang Mao, and Honglan Jiang. An architectural error metric for CNN-oriented approximate multipliers. *arXiv preprint arXiv:2408.12836*, 2024. doi: 10.48550/arXiv.2408.12836. URL https://arxiv.org/abs/2408.12836.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. doi: 10.1109/ICCV48922.2021.00986.

Chang Meng, Wayne P. Burleson, Weikang Qian, and Giovanni De Micheli. Gradient approximation of approximate multipliers for high-accuracy deep neural network retraining. In *Design, Automation and Test in Europe Conference (DATE)*, pp. 1–7, 2025. doi: 10.23919/DATE64628.2025.10992942.

Chang Meng, Hanyu Wang, Yuyang Ye, Mingfei Yu, Wayne Burleson, and Giovanni De Micheli. TRAM: Training approximate multiplier structures for low-power AI accelerators. *arXiv preprint arXiv:2605.08231*, 2026. doi: 10.48550/arXiv.2605.08231. URL https://arxiv.org/abs/2605.08231.

Hai Mo, Yong Wu, Honglan Jiang, Zining Ma, Fabrizio Lombardi, Jie Han, and Leibo Liu. Learning the error features of approximate multipliers for neural network applications. *IEEE Transactions on Computers*, 73(3):842–856, 2024. doi: 10.1109/TC.2023.3345163.

Vojtech Mrazek, Zdenek Vasicek, Lukas Sekanina, Muhammad Abdullah Hanif, and Muhammad Shafique. ALWANN: Automatic layer-wise approximation of deep neural network accelerators without retraining. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019a. doi: 10.1109/ICCAD45719.2019.8942068.

Vojtech Mrazek, Zdenek Vasicek, Lukas Sekanina, and Muhammad Shafique. autoAx: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components. In *Proceedings of the 56th Annual Design Automation Conference (DAC)*, 2019b.

Aikaterini Maria Panteleaki, Konstantinos Balaskas, Georgios Zervakis, Hussam Amrouch, and Iraklis Anagnostopoulos. Late breaking results: Leveraging approximate computing for carbon-aware DNN accelerators. In *Design, Automation & Test in Europe Conference (DATE)*, pp. 1–2, 2025. doi: 10.23919/DATE64628.2025.10993191.

Michal Pinos, Lukas Sekanina, and Vojtech Mrazek. ApproxDARTS: Differentiable neural architecture search with approximate multipliers. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2024. doi: 10.1109/IJCNN60899.2024.10650823.

Flavio Ponzina, Giovanni Ansaloni, Miguel Peón-Quirós, and David Atienza. Using algorithmic transformations and sensitivity analysis to unleash approximations in CNNs at the edge. *Micromachines*, 13(7):1143, 2022. doi: 10.3390/mi13071143.

Chirag Sakhuja, Zhan Shi, and Calvin Lin. Leveraging domain information for the efficient automated design of deep learning accelerators. In *Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 287–301, 2023. doi: 10.1109/HPCA56546.2023.10071095.

Chirag Sakhuja, Charles Hong, and Calvin Lin. Polaris: Multi-fidelity design space exploration of deep learning accelerators. *arXiv preprint arXiv:2412.15548*, 2024. doi: 10.48550/arXiv.2412.15548. URL https://arxiv.org/abs/2412.15548.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pp. 2951–2959, 2012.

Ourania Spantidi, Georgios Zervakis, Iraklis Anagnostopoulos, Hussam Amrouch, and Jörg Henkel. Positive/negative approximate multipliers for DNN accelerators. In *Proceedings of the 2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2021. doi: 10.1109/ICCAD51958.2021.9643491.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Salim Ullah, Siva Satyendra Sahoo, Nemath Ahmed, Debabrata Chaudhury, and Akash Kumar. AppAxO: Designing application-specific approximate operators for FPGA-based embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 21(3):1–31, 2022. doi: 10.1145/3513262.

Ondrej Vlcek and Vojtech Mrazek. ApproxGNN: A pretrained GNN for parameter prediction in design space exploration for approximate computing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025. doi: 10.1109/ICCAD66269.2025.11240776.

Ying Wu, Chuangtao Chen, Weihua Xiao, Xuan Wang, Chenyi Wen, Jie Han, Xunzhao Yin, Weikang Qian, and Cheng Zhuo. A survey on approximate multiplier designs for energy efficiency: From algorithms to circuits. *ACM Transactions on Design Automation of Electronic Systems*, 29(1):1–37, 2024. doi: 10.1145/3610291.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Tongxin Yang, Tomoaki Ukezono, and Toshinori Sato. Design and analysis of a low-power high-speed accuracy-controllable approximate multiplier. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E101-A(12):2244–2253, 2018. doi: 10.1587/transfun.E101.A.2244.

Tianyang Yu, Bi Wu, Ke Chen, Chenggang Yan, and Weiqiang Liu. Toward efficient retraining: A large-scale approximate neural network framework with cross-layer optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 32(6):1004–1017, 2024. doi: 10.1109/TVLSI.2024.3386900.

Su Zheng, Zhen Li, Yao Lu, Jingbo Gao, Jide Zhang, and Lingli Wang. HEAM: High-efficiency approximate multiplier optimization for deep neural networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 3359–3363, 2022. doi: 10.1109/ISCAS48785.2022.9937321.

Yiqi Zhou, JunHao Ma, Xingyang Li, Yule Sheng, Yue Yuan, Yikai Wang, Bochang Wang, Yiheng Wu, Shan Shen, Wei Xing, Daying Sun, Li Li, and Zhiqiang Xiao. OpenACM: An open-source SRAM-based approximate CiM compiler. In *Proceedings of the IEEE/ACM Design, Automation and Test in Europe Conference (DATE)*, 2026a.

Yiqi Zhou, Yue Yuan, Yikai Wang, Bohao Liu, Qinxin Mei, Zhuohua Liu, Shan Shen, Wei Xing, Daying Sun, Li Li, and Guozhu Liu. OpenACMv2: An accuracy-constrained co-optimization framework for approximate DCiM. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, 2026b. doi: 10.1145/3770743.3804259.

A Related Work

A.1 Approximate Computing for Efficient Neural Inference

Approximate computing has been widely explored to improve neural-inference efficiency by trading computational accuracy for reduced hardware cost. Existing approximate multipliers employ techniques such as operand truncation, approximate compressors, partial-product reduction, and error compensation, and are commonly evaluated using circuit-level error and PPA metrics (Yang et al., 2018; Armeniakos et al., 2022; Hu et al., 2024; Wu et al., 2024). Automated generation and design-space exploration frameworks further enable large-scale construction and optimization of approximate multipliers (Mrazek et al., 2019b; Zhou et al., 2026a; Meng et al., 2026), while model adaptation and retraining techniques improve their deployment in neural networks (Danopoulos et al., 2023; Gong et al., 2023; Yu et al., 2024; Meng et al., 2025). However, these methods primarily address multiplier generation, deployment, or adaptation, while efficiently evaluating the application-level accuracy impact of newly generated designs remains challenging.

A.2 Application-Aware Error Tolerance in Approximate Computing

Application-aware error tolerance characterizes how the error behavior of an approximate multiplier translates into accuracy degradation for a specific neural model. The same multiplier can exhibit different impacts across models because arithmetic errors interact differently with model parameters, activation distributions, and computation patterns (Armeniakos et al., 2022; Leon et al., 2025; Wu et al., 2024). Conventional metrics such as mean error distance (MED), normalized mean error distance (NMED), mean relative error distance (MRED), error rate (ER), and worst-case error (WCE) provide efficient summaries of circuit-level error but discard the structured distribution of errors across the input space (Yang et al., 2018; Wu et al., 2024). Learning-based methods predict application-level accuracy from multiplier-level or statistical error features (Ansari et al., 2020; Mo et al., 2024), but selected statistics cannot fully preserve the underlying error structure. AME instead combines the complete multiplier error matrix with model-dependent activation and weight information to estimate CNN accuracy degradation, explicitly accounting for workload dependence but relying on analytical error-propagation assumptions (Liu et al., 2024). Existing approaches therefore leave room for application-aware prediction that preserves structured multiplier error information while accounting for model-specific execution characteristics. Table 5 summarizes how representative methods for application-level impact estimation differ in their inputs and in the model families they cover.

Table 5: Comparison of Methods for Estimating the Application-Level Impact of Approximate Arithmetic

Method	**Input Features**	**Architecture**
Statistical Features (Ansari et al., 2020)	Statistical multiplier features	CNN
Statistical Error Features (Mo et al., 2024)	Selected statistical error features	CNN
AME (Liu et al., 2024)	Full error matrix; activation and weight distributions	CNN
AAAE (Ours)	Multi-scale error surface; measured input-value distribution; labeled probe response	Transformer; CNN variant

A.3 Accuracy-Constrained Design Space Exploration for Approximate Computing

Accuracy-constrained design-space exploration seeks hardware-efficient approximate computing designs while maintaining application-level accuracy requirements. Existing methods optimize the assignment or configuration of predefined approximate multipliers across neural-network components (Mrazek et al., 2019a; Jain et al., 2022; Pinos et al., 2024; Danopoulos et al., 2025), while other approaches incorporate application information into approximate multiplier generation and

optimization (Zheng et al., 2022; Ullah et al., 2022). Surrogate models and multi-fidelity optimization further reduce expensive evaluations during large-scale design-space exploration (Mrazek et al., 2019b; Sakhuja et al., 2023; 2024; Vlcek & Mrazek, 2025; Zhou et al., 2026b). A remaining challenge is to translate model-specific error tolerance into an effective design signal for approximate multiplier optimization. This motivates accuracy-aware methods that characterize how multiplier error patterns interact with neural models and use the resulting accuracy predictions to guide the search toward hardware-efficient multipliers that conform to model-specific tolerance.

B Experimental Details and Implementation Validation

B.1 Accuracy-Drop Predictor Schemas

The persisted static schema of $s(a)$ contains 3072 projection values and 31 descriptors: each of the four channels in equation 5 contributes 256 row, 256 column, and 256 grid values, and the descriptor block contains the 19-entry generator record d_a together with 12 global error statistics (signed and absolute means and standard deviations, relative-error aggregates over nonzero products, RMSE, RRMSE, and NMED normalized by the largest absolute exact product). A signed-log map is applied to every projection and descriptor, so $s(a) \in \mathbb{R}^{3103}$.

B.2 Transformer Aggregation Schema

The Transformer encoder groups the propagated contributions of equation 8 by block and operation role, forming the per-block sums and upstream prefixes

$$C_{m,a,i,b} = \sum_{o} \xi_{m,a,i,(b,o)}, \qquad P_{m,a,i,b} = \sum_{t=1}^{b} C_{m,a,i,t}, \tag{11}$$

with $P_{m,a,i,0} = 0$. Path summaries add absolute, positive, negative, and squared magnitudes, a cancellation ratio against the full-depth prefix $P_{m,a,i,12}$, and the block-prefix interaction $Q_{m,a,i,b} = C_{m,a,i,b}P_{m,a,i,b-1}$. For each depth segment d, AAAE records the soft boundary response

$$B_\zeta(m,a,i,d) = \sigma\left(-\frac{M_i + P^{(d)}_{m,a,i}}{\zeta}\right) - \sigma\left(-\frac{M_i}{\zeta}\right) \tag{12}$$

at six fixed scales ζ, together with a hard-crossing indicator and the normalized shift $P^{(d)}_{m,a,i}/(M_i + 0.05)$, whose distribution over the 256 profile images is summarized by 11 quantiles. Five fixed, weight-normalized image groups cover all, clean-wrong, and clean-correct images at low, middle, or high clean margins; Swin uses four architectural stages, and ViT and DeiT use four three-block depth segments. The schema allocates 1034 entries to these summaries, so $c_{\mathrm{T}}(m,a) \in \mathbb{R}^{1034}$.

B.3 Response Schema and Training Protocol

The response encoder retains the six ordered length-50 arrays $\widetilde{b}^{(1)}$, $\widetilde{b}^{(5)}$, $F^{(1)}$, $F^{(5)}$, $\Delta M^{(1)}$, and $\Delta M^{(5)}$, and computes for each correctness or margin-change array the mean, population standard deviation, mean absolute value, positive- and negative-part means, and nine quantiles, yielding $r_{\mathrm{obs},50}(m,a) \in \mathbb{R}^{358}$. Two static LightGBM priors $\pi^{(k)}_{m,a} = h_{m,k}(s(a))$ add the residuals $\bar{d}^{(k)}_a - \pi^{(k)}_{m,a}$, so $r_{50} \equiv r_{\mathrm{hyb},50} \in \mathbb{R}^{360}$; response-only and leave-s-out ablations use $r_{\mathrm{obs},50}$ instead.

The greedy selector starts from $S_0 = \emptyset$. At step t, it evaluates each remaining image j through its extension error $e^{(k)}_{a,t}(j) = \widehat{d}^{(k)}_{a,t}(j) - \bar{y}^{(k)}_{m,a}$ against full-evaluation drops on training designs, where $\bar{y}_{m,a} = y_{m,a}/100$ is the proportion-scale drop and $\widehat{d}^{(k)}_{a,t}(j)$ is the running mean of $F^{(k)}_{a,\cdot}$ over $S_{t-1} \cup \{j\}$, and minimizes

$$J_t(j) = \sum_{k\in\{1,5\}} \left[\mathrm{MAE}_a(e^{(k)}_{a,t}(j)) + \lambda_{\mathrm{tail}} Q_{.95}(|e^{(k)}_{a,t}(j)|) + \lambda_{\max} \max_a |e^{(k)}_{a,t}(j)|\right], \tag{13}$$

with $\lambda_{\mathrm{tail}} = 0.2$ and $\lambda_{\max} = 0.05$. AAAE adds the image with the smallest $J_t(j)$, finalizes S_{50} and its order, and then constructs validation, calibration, and test inputs.

Table 6: Implemented accuracy-prediction paths by model family. The response column reports additional model inference beyond the precomputed clean-model profile.

Family	Predictor inputs	Prediction head	Response
ViT / DeiT / Swin	3103 static +360 hybrid response +1034 conditioned = 4497	Per-target-model LightGBM	50 labeled images
ResNet-50	3185 engineered values +2 × 129 × 129 residual LUT	Two-member Delta-CNN bundle	N/A
MobileNetV3-Large	3233 engineered values +2 × 129 × 129 residual LUT	Two-member Correctness-CNN bundle with disagreement	N/A

After model selection, AAAE refits the selected Transformer mapping on the 17,000-design training–validation union by minimizing squared prediction error plus a GBDT regularizer, with out-of-fold prior predictions populating training and refit rows, a training-only prior populating validation rows, and deployment priors fit on the complete refit set before generating calibration, test, and deployment inputs. The fixed output clipping is $\mathcal{C}(x) = \mathrm{clip}(x, -0.05, 1)$.

B.4 CNN Application-Conditioned Predictor

For convolutional layer ℓ, reuse-weighted activation and weight histograms $p_{m,\ell}(u)$ and $q_{m,\ell}(v)$, each normalized to one, describe the factorized input distribution, and the layer MAC count $N_{m,\ell}^{\mathrm{MAC}}$ with an RMS $\pm\epsilon$ probe response defines a per-effective-MAC propagation gain $\gamma_{m,\ell}$; layers without nonzero input mass use a zero gain and a required zero probe response. These per-layer gains form an application-weighted error feature block that is standardized with training-set statistics and clipped to $[-12, 12]$, and the neural head additionally reads a two-channel $2 \times 129 \times 129$ residual-LUT encoding, $I_{a,1} = \mathrm{sgn}(E_a^{\square}) \log(1 + |E_a^{\square}|)/\log(1025)$ and $I_{a,2} = \log(1 + |E_a^{\square}|)/\log(1025)$, over the two's-complement-ordered surface $E_a^{\square}$.

ResNet-50 uses a two-member Delta-CNN predictor that regresses $\bar{y}_{m,a}$ directly, while MobileNetV3-Large uses two Correctness-CNN members that aggregate per-image correctness logits into a drop estimate and are trained with a weighted binary cross-entropy loss plus squared-error and top-K tail terms, where $K = \lceil q|\mathcal{B}| \rceil$ selects the K largest squared errors of the minibatch. Each bundle combines its two members through a validation-fitted linear combination with an optional disagreement term; ResNet-50 sets the disagreement coefficient to zero.

B.5 Evaluation Protocol Details

The paired Top-1 drop uses the fixed evaluation set $\mathcal{D}_{\mathrm{eval}}$ and the exact quantized baseline, and prediction error is reported in percentage points. We record the identities and overlap of the profile, response, and evaluation images; CNNs have no candidate-level response set. A separate audit set $\mathcal{D}_{\mathrm{audit}}$ is used only after selection. Screening results are reported with accepted and feasible denominators, and hardware results use PDP under a specified circuit flow. The split manifests are given in Section B.6.

B.6 Candidate Pool and Split Manifests

The original multiplier pool contains 20,000 designs, partitioned into 14,000 Train, 3,000 Validation, and 3,000 Test configurations; the configuration hash, the low-token sequence, and the runtime design ID each take 20,000 distinct values, so no duplicate structure was removed. The 3,000 Calibration candidates come from a separately generated pool.

B.7 Metric Definitions

For a target model m and candidate design a, let $\widehat{y}_{m,a}$ be the predicted and $y_{m,a}$ the measured Top-1 drop, and let $e_{m,a} = \widehat{y}_{m,a} - y_{m,a}$ denote the prediction error in percentage points. Over the n

candidates of a test set, the prediction-accuracy metrics reported in Tables 1 and 2 are

$$\mathrm{MAE} = \frac{1}{n}\sum_{a}|e_{m,a}|, \qquad \mathrm{P95} = Q_{0.95}\big(\{|e_{m,a}|\}\big), \qquad \mathrm{MaxAE} = \max_{a}|e_{m,a}|, \tag{14}$$

where $Q_{0.95}$ denotes the empirical 95th percentile, and Spearman ρ is the Pearson correlation between the ranks of $\{\widehat{y}_{m,a}\}$ and $\{y_{m,a}\}$. For the hardware-cost predictor of Table 4, with per-metric errors e_i and targets y_i over n test multipliers,

$$\mathrm{MSE} = \frac{1}{n}\sum_{i} e_i^2, \qquad \mathrm{MRE} = \frac{1}{n}\sum_{i}\frac{|e_i|}{|y_i|}, \qquad R^2 = 1 - \frac{\sum_i (y_i - \widehat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}. \tag{15}$$

For the screening study of Table 3, with the admission bound $U^{(\alpha)}_{m,a}$ of equation 10, budget τ, and candidate pool $\mathcal{C}$ of size n,

$$\mathrm{Cov} = \frac{\#\{a \in \mathcal{C} : y_{m,a} \le U^{(\alpha)}_{m,a}\}}{n}, \tag{16}$$

$$\mathrm{FDP} = \frac{\#\{a \in \mathcal{C} : y_{m,a} > \tau \wedge U^{(\alpha)}_{m,a} \le \tau\}}{\#\{a \in \mathcal{C} : U^{(\alpha)}_{m,a} \le \tau\}}, \qquad \mathrm{Rec} = \frac{\#\{a \in \mathcal{C} : y_{m,a} \le \tau \wedge U^{(\alpha)}_{m,a} \le \tau\}}{\#\{a \in \mathcal{C} : y_{m,a} \le \tau\}}, \tag{17}$$

and the power–delay product is $\mathrm{PDP}(a) = \mathrm{Power}(a) \times \mathrm{Delay}(a)$. In Table 3 coverage is a fraction, while FDP and feasible recall are percentages, and only A1/A2 are calibrated bounds with nominal coverage 0.90.

C Calibration Diagnostics and Optimization Failure Analysis

For Calibration design i, let $b(i)$ be its predefined prediction bin, and let $\widehat{\pi}_{\mathrm{cal},b}$ and $\widehat{\pi}_{\mathcal{C},b}$ be the fractions of Calibration and target-pool designs in bin b. With a fixed smoothing constant $\epsilon > 0$, the one-sided underprediction score and the pool weight are

$$s^{+}_{m,i} = \max\{y_{m,i} - 100\widehat{y}_{m,i}, 0\}, \qquad w_i = \frac{\widehat{\pi}_{\mathcal{C},b(i)} + \epsilon}{\widehat{\pi}_{\mathrm{cal},b(i)} + \epsilon}. \tag{18}$$

The importance-weighted empirical $(1-\alpha)$ quantile of these scores is the smallest t with $\sum_i w_i \mathbf{1}\{s^{+}_{m,i} \le t\}/\sum_i w_i \ge 1-\alpha$, giving the correction $q^{(\mathcal{C})}_{m,\alpha}$ of Section 2.4. A1 applies this correction uniformly to every candidate, while A2 is its bin-wise (Mondrian) variant: the $(1-\alpha)$ quantile of the same scores is recomputed inside each prediction bin b, so designs in different bins receive different corrections. A different candidate pool requires updated bin weights, and the correction provides neither a standard finite-sample split-conformal guarantee nor control of the false-discovery proportion among admitted designs.

Tables 7–11 report the complete calibrated-screening counts for all five models, the four admission rules, every threshold τ, and every calibration level α; the $\tau = 0.5$ rows are the label-resolution stress test.

Table 7: Calibrated-screening counts for Swin-Tiny ($n = 3{,}000$ Test candidates): coverage $\#(y \leq U)/n$, admission $\#(U \leq \tau)/n$, false-discovery $\#(y > \tau \wedge U \leq \tau)/\#(U \leq \tau)$, and feasible recall $\#(y \leq \tau \wedge U \leq \tau)/\#(y \leq \tau)$. Rows with a dash are independent of α; N/A marks an empty denominator.

Rule	α	τ (%)	Cov. $\#/n$	Adm. $\#/n$	FDP $\#/\#$	Recall $\#/\#$
Point	–	0.5	1497/3000	0/3000	N/A	0/15
Point	–	1.0	1497/3000	27/3000	8/27	19/59
Point	–	2.0	1497/3000	211/3000	23/211	188/232
Point	–	5.0	1497/3000	663/3000	23/663	640/669
Fixed-Margin	–	0.5	2784/3000	0/3000	N/A	0/15
Fixed-Margin	–	1.0	2784/3000	0/3000	N/A	0/59
Fixed-Margin	–	2.0	2784/3000	27/3000	0/27	27/232
Fixed-Margin	–	5.0	2784/3000	560/3000	1/560	559/669
A1	0.10	0.5	2661/3000	0/3000	N/A	0/15
A1	0.10	1.0	2661/3000	0/3000	N/A	0/59
A1	0.10	2.0	2661/3000	57/3000	0/57	57/232
A1	0.10	5.0	2661/3000	586/3000	4/586	582/669
A1	0.05	0.5	2839/3000	0/3000	N/A	0/15
A1	0.05	1.0	2839/3000	0/3000	N/A	0/59
A1	0.05	2.0	2839/3000	9/3000	0/9	9/232
A1	0.05	5.0	2839/3000	543/3000	0/543	543/669
A1	0.01	0.5	2985/3000	0/3000	N/A	0/15
A1	0.01	1.0	2985/3000	0/3000	N/A	0/59
A1	0.01	2.0	2985/3000	0/3000	N/A	0/232
A1	0.01	5.0	2985/3000	339/3000	0/339	339/669
A2	0.10	0.5	2666/3000	0/3000	N/A	0/15
A2	0.10	1.0	2666/3000	0/3000	N/A	0/59
A2	0.10	2.0	2666/3000	134/3000	4/134	130/232
A2	0.10	5.0	2666/3000	595/3000	4/595	591/669
A2	0.05	0.5	2859/3000	0/3000	N/A	0/15
A2	0.05	1.0	2859/3000	0/3000	N/A	0/59
A2	0.05	2.0	2859/3000	105/3000	0/105	105/232
A2	0.05	5.0	2859/3000	568/3000	1/568	567/669
A2	0.01	0.5	2984/3000	0/3000	N/A	0/15
A2	0.01	1.0	2984/3000	0/3000	N/A	0/59
A2	0.01	2.0	2984/3000	40/3000	0/40	40/232
A2	0.01	5.0	2984/3000	523/3000	0/523	523/669

Table 8: Calibrated-screening counts for ViT-Base ($n = 3{,}000$ Test candidates): coverage $\#(y \leq U)/n$, admission $\#(U \leq \tau)/n$, false-discovery $\#(y > \tau \wedge U \leq \tau)/\#(U \leq \tau)$, and feasible recall $\#(y \leq \tau \wedge U \leq \tau)/\#(y \leq \tau)$. Rows with a dash are independent of α; N/A marks an empty denominator.

Rule	α	τ (%)	Cov. $\#/n$	Adm. $\#/n$	FDP $\#/\#$	Recall $\#/\#$
Point	–	0.5	1458/3000	18/3000	7/18	11/44
Point	–	1.0	1458/3000	93/3000	13/93	80/114
Point	–	2.0	1458/3000	289/3000	47/289	242/277
Point	–	5.0	1458/3000	701/3000	14/701	687/718
Fixed-Margin	–	0.5	2741/3000	0/3000	N/A	0/44
Fixed-Margin	–	1.0	2741/3000	0/3000	N/A	0/114
Fixed-Margin	–	2.0	2741/3000	93/3000	0/93	93/277
Fixed-Margin	–	5.0	2741/3000	611/3000	0/611	611/718
A1	0.10	0.5	2677/3000	0/3000	N/A	0/44
A1	0.10	1.0	2677/3000	0/3000	N/A	0/114
A1	0.10	2.0	2677/3000	114/3000	1/114	113/277
A1	0.10	5.0	2677/3000	629/3000	1/629	628/718
A1	0.05	0.5	2856/3000	0/3000	N/A	0/44
A1	0.05	1.0	2856/3000	0/3000	N/A	0/114
A1	0.05	2.0	2856/3000	51/3000	0/51	51/277
A1	0.05	5.0	2856/3000	586/3000	0/586	586/718
A1	0.01	0.5	2974/3000	0/3000	N/A	0/44
A1	0.01	1.0	2974/3000	0/3000	N/A	0/114
A1	0.01	2.0	2974/3000	0/3000	N/A	0/277
A1	0.01	5.0	2974/3000	393/3000	0/393	393/718
A2	0.10	0.5	2559/3000	0/3000	N/A	0/44
A2	0.10	1.0	2559/3000	6/3000	0/6	6/114
A2	0.10	2.0	2559/3000	93/3000	0/93	93/277
A2	0.10	5.0	2559/3000	614/3000	1/614	613/718
A2	0.05	0.5	2689/3000	0/3000	N/A	0/44
A2	0.05	1.0	2689/3000	0/3000	N/A	0/114
A2	0.05	2.0	2689/3000	34/3000	0/34	34/277
A2	0.05	5.0	2689/3000	573/3000	0/573	573/718
A2	0.01	0.5	2942/3000	0/3000	N/A	0/44
A2	0.01	1.0	2942/3000	0/3000	N/A	0/114
A2	0.01	2.0	2942/3000	29/3000	0/29	29/277
A2	0.01	5.0	2942/3000	382/3000	0/382	382/718

Table 9: Calibrated-screening counts for DeiT-Base ($n = 3{,}000$ Test candidates): coverage $\#(y \leq U)/n$, admission $\#(U \leq \tau)/n$, false-discovery $\#(y > \tau \wedge U \leq \tau)/\#(U \leq \tau)$, and feasible recall $\#(y \leq \tau \wedge U \leq \tau)/\#(y \leq \tau)$. Rows with a dash are independent of α; N/A marks an empty denominator.

Rule	α	τ (%)	Cov. $\#/n$	Adm. $\#/n$	FDP $\#/\#$	Recall $\#/\#$
Point	–	0.5	1421/3000	523/3000	149/523	374/529
Point	–	1.0	1421/3000	876/3000	86/876	790/828
Point	–	2.0	1421/3000	1124/3000	35/1124	1089/1098
Point	–	5.0	1421/3000	1346/3000	9/1346	1337/1342
Fixed-Margin	–	0.5	2860/3000	0/3000	N/A	0/529
Fixed-Margin	–	1.0	2860/3000	0/3000	N/A	0/828
Fixed-Margin	–	2.0	2860/3000	876/3000	1/876	875/1098
Fixed-Margin	–	5.0	2860/3000	1292/3000	0/1292	1292/1342
A1	0.10	0.5	2482/3000	0/3000	N/A	0/529
A1	0.10	1.0	2482/3000	572/3000	20/572	552/828
A1	0.10	2.0	2482/3000	1027/3000	10/1027	1017/1098
A1	0.10	5.0	2482/3000	1330/3000	5/1330	1325/1342
A1	0.05	0.5	2705/3000	0/3000	N/A	0/529
A1	0.05	1.0	2705/3000	197/3000	3/197	194/828
A1	0.05	2.0	2705/3000	924/3000	2/924	922/1098
A1	0.05	5.0	2705/3000	1311/3000	2/1311	1309/1342
A1	0.01	0.5	2911/3000	0/3000	N/A	0/529
A1	0.01	1.0	2911/3000	0/3000	N/A	0/828
A1	0.01	2.0	2911/3000	810/3000	0/810	810/1098
A1	0.01	5.0	2911/3000	1274/3000	0/1274	1274/1342
A2	0.10	0.5	2437/3000	8/3000	0/8	8/529
A2	0.10	1.0	2437/3000	550/3000	18/550	532/828
A2	0.10	2.0	2437/3000	981/3000	10/981	971/1098
A2	0.10	5.0	2437/3000	1329/3000	5/1329	1324/1342
A2	0.05	0.5	2657/3000	0/3000	N/A	0/529
A2	0.05	1.0	2657/3000	281/3000	6/281	275/828
A2	0.05	2.0	2657/3000	930/3000	3/930	927/1098
A2	0.05	5.0	2657/3000	1314/3000	4/1314	1310/1342
A2	0.01	0.5	2910/3000	0/3000	N/A	0/529
A2	0.01	1.0	2910/3000	5/3000	0/5	5/828
A2	0.01	2.0	2910/3000	785/3000	1/785	784/1098
A2	0.01	5.0	2910/3000	1282/3000	0/1282	1282/1342

Table 10: Calibrated-screening counts for ResNet-50 ($n = 3{,}000$ Test candidates): coverage $\#(y \leq U)/n$, admission $\#(U \leq \tau)/n$, false-discovery $\#(y > \tau \wedge U \leq \tau)/\#(U \leq \tau)$, and feasible recall $\#(y \leq \tau \wedge U \leq \tau)/\#(y \leq \tau)$. Rows with a dash are independent of α; N/A marks an empty denominator.

Rule	α	τ (%)	Cov. $\#/n$	Adm. $\#/n$	FDP $\#/\#$	Recall $\#/\#$
Point	–	0.5	2382/3000	0/3000	N/A	0/407
Point	–	1.0	2382/3000	45/3000	0/45	45/736
Point	–	2.0	2382/3000	890/3000	2/890	888/1053
Point	–	5.0	2382/3000	2045/3000	11/2045	2034/2075
Fixed-Margin	–	0.5	2917/3000	0/3000	N/A	0/407
Fixed-Margin	–	1.0	2917/3000	0/3000	N/A	0/736
Fixed-Margin	–	2.0	2917/3000	45/3000	0/45	45/1053
Fixed-Margin	–	5.0	2917/3000	1834/3000	0/1834	1834/2075
A1	0.10	0.5	2894/3000	0/3000	N/A	0/407
A1	0.10	1.0	2894/3000	0/3000	N/A	0/736
A1	0.10	2.0	2894/3000	237/3000	0/237	237/1053
A1	0.10	5.0	2894/3000	1871/3000	0/1871	1871/2075
A1	0.05	0.5	2970/3000	0/3000	N/A	0/407
A1	0.05	1.0	2970/3000	0/3000	N/A	0/736
A1	0.05	2.0	2970/3000	0/3000	N/A	0/1053
A1	0.05	5.0	2970/3000	1476/3000	0/1476	1476/2075
A1	0.01	0.5	2997/3000	0/3000	N/A	0/407
A1	0.01	1.0	2997/3000	0/3000	N/A	0/736
A1	0.01	2.0	2997/3000	0/3000	N/A	0/1053
A1	0.01	5.0	2997/3000	889/3000	0/889	889/2075
A2	0.10	0.5	2759/3000	0/3000	N/A	0/407
A2	0.10	1.0	2759/3000	44/3000	0/44	44/736
A2	0.10	2.0	2759/3000	839/3000	2/839	837/1053
A2	0.10	5.0	2759/3000	1813/3000	5/1813	1808/2075
A2	0.05	0.5	2805/3000	0/3000	N/A	0/407
A2	0.05	1.0	2805/3000	39/3000	0/39	39/736
A2	0.05	2.0	2805/3000	836/3000	1/836	835/1053
A2	0.05	5.0	2805/3000	1753/3000	4/1753	1749/2075
A2	0.01	0.5	2957/3000	0/3000	N/A	0/407
A2	0.01	1.0	2957/3000	14/3000	0/14	14/736
A2	0.01	2.0	2957/3000	813/3000	1/813	812/1053
A2	0.01	5.0	2957/3000	1600/3000	0/1600	1600/2075

Table 11: Calibrated-screening counts for MobileNetV3-Large ($n = 3{,}000$ Test candidates): coverage $\#(y \le U)/n$, admission $\#(U \le \tau)/n$, false-discovery $\#(y > \tau \wedge U \le \tau)/\#(U \le \tau)$, and feasible recall $\#(y \le \tau \wedge U \le \tau)/\#(y \le \tau)$. Rows with a dash are independent of α; N/A marks an empty denominator.

Rule	α	τ (%)	Cov. $\#/n$	Adm. $\#/n$	FDP $\#/\#$	Recall $\#/\#$
Point	–	0.5	1507/3000	133/3000	38/133	95/105
Point	–	1.0	1507/3000	179/3000	34/179	145/154
Point	–	2.0	1507/3000	287/3000	46/287	241/251
Point	–	5.0	1507/3000	532/3000	17/532	515/538
Fixed-Margin	–	0.5	2854/3000	0/3000	N/A	0/105
Fixed-Margin	–	1.0	2854/3000	51/3000	2/51	49/154
Fixed-Margin	–	2.0	2854/3000	179/3000	1/179	178/251
Fixed-Margin	–	5.0	2854/3000	457/3000	1/457	456/538
A1	0.10	0.5	2926/3000	0/3000	N/A	0/105
A1	0.10	1.0	2926/3000	10/3000	0/10	10/154
A1	0.10	2.0	2926/3000	160/3000	1/160	159/251
A1	0.10	5.0	2926/3000	441/3000	1/441	440/538
A1	0.05	0.5	2974/3000	0/3000	N/A	0/105
A1	0.05	1.0	2974/3000	0/3000	N/A	0/154
A1	0.05	2.0	2974/3000	118/3000	0/118	118/251
A1	0.05	5.0	2974/3000	408/3000	0/408	408/538
A1	0.01	0.5	2994/3000	0/3000	N/A	0/105
A1	0.01	1.0	2994/3000	0/3000	N/A	0/154
A1	0.01	2.0	2994/3000	0/3000	N/A	0/251
A1	0.01	5.0	2994/3000	299/3000	0/299	299/538
A2	0.10	0.5	2750/3000	0/3000	N/A	0/105
A2	0.10	1.0	2750/3000	6/3000	0/6	6/154
A2	0.10	2.0	2750/3000	146/3000	0/146	146/251
A2	0.10	5.0	2750/3000	437/3000	0/437	437/538
A2	0.05	0.5	2867/3000	0/3000	N/A	0/105
A2	0.05	1.0	2867/3000	0/3000	N/A	0/154
A2	0.05	2.0	2867/3000	105/3000	0/105	105/251
A2	0.05	5.0	2867/3000	419/3000	0/419	419/538
A2	0.01	0.5	2997/3000	0/3000	N/A	0/105
A2	0.01	1.0	2997/3000	0/3000	N/A	0/154
A2	0.01	2.0	2997/3000	13/3000	0/13	13/251
A2	0.01	5.0	2997/3000	336/3000	0/336	336/538

D Extended Multiplier Design Optimization Results

The upper bound of Section 2.4 determines accuracy admission, while predicted PDP orders admitted designs for at most K full-model accuracy queries:

$$\mathcal{A}_{\mathrm{adm}} = \{a \in \mathcal{C} : U^{(\alpha)}_{m,a} \le \tau\}, \qquad \mathcal{S}_K = \mathrm{TopK}_{\uparrow \widehat{h}}(\mathcal{A}_{\mathrm{adm}}). \tag{19}$$

PDP ties are broken by configuration hash, and $\min\{K, |\mathcal{A}_{\mathrm{adm}}|\}$ designs are queried with the 1,000-image measurement of Section 2.2.5; the Test study replays these queries from precomputed labels. Within this protocol, a queried design is feasible when $y_{m,a} \le \tau$, and one run selects

$$\mathcal{F}_K = \{a \in \mathcal{S}_K : y_{m,a} \le \tau\}, \qquad a^\dagger = \arg\min_{a \in \mathcal{F}_K} \widehat{h}(a), \qquad h^{\mathrm{ver}}_{a^\dagger} = P^{\mathrm{ver}}_{a^\dagger} D^{\mathrm{ver}}_{a^\dagger}. \tag{20}$$

An empty $\mathcal{F}_K$ is a failed selection. Same-image reruns, independent image audits, and the reviewed-set measured front remain separate checks; the front aggregates selected and diagnostic points across runs, and none changes the single-run choice.

Figure 5 reports the NSGA-II run of the three-objective search on the same 20,000-candidate pool, drawn with the same projections and the same predicted-versus-verified convention as Figure 4. Compared at matched Top-1 drop (0.7–49 %), the verified front again lies below the predicted one (median 2.0 fJ in PDP and 2.1 μm^2 in area), and 11 verified designs fall inside the $\tau = 2$ % budget; the lowest verified PDP and area among them are 222.7 fJ and 464.4 μm^2, respectively.

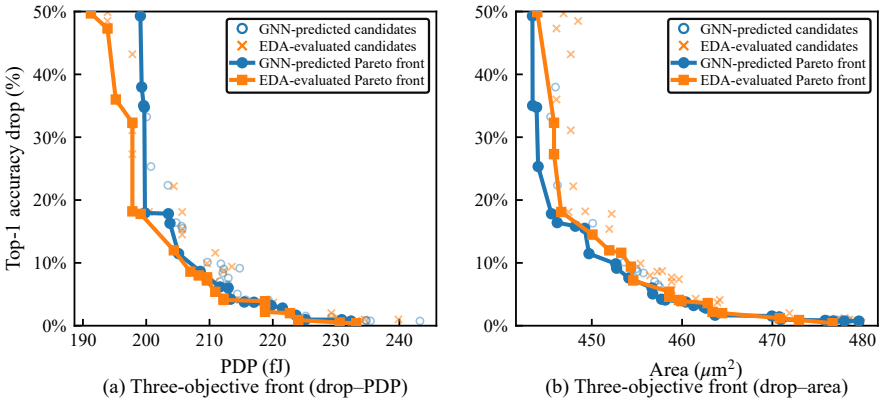


Figure 5: NSGA-II run of the three-objective search of Section 3.6 on the 20,000-candidate pool, shown with the same projections as Figure 4. Faded markers are candidate designs and bold curves the Pareto fronts; blue is GNN-predicted and orange EDA-verified. The ordinate is the Top-1 drop in percent, and candidates above a 50 % drop are omitted.

Figure 6 shows the earlier benchmark, which is bi-objective: the search optimised Top-1 drop and PDP only, on a 30,000-candidate pool, and no area objective or area constraint was active. Its fronts therefore describe a different problem and are not directly comparable with the three-objective fronts above; they are reported only as a reference point. Within their own setting the verified fronts again stay at or below the predicted ones at matched Top-1 drop (median 2.4 fJ for both searches), and the cheapest verified design inside the $\tau = 2$ % budget reaches 208.3 fJ for Bayesian optimization and 222.7 fJ for NSGA-II.

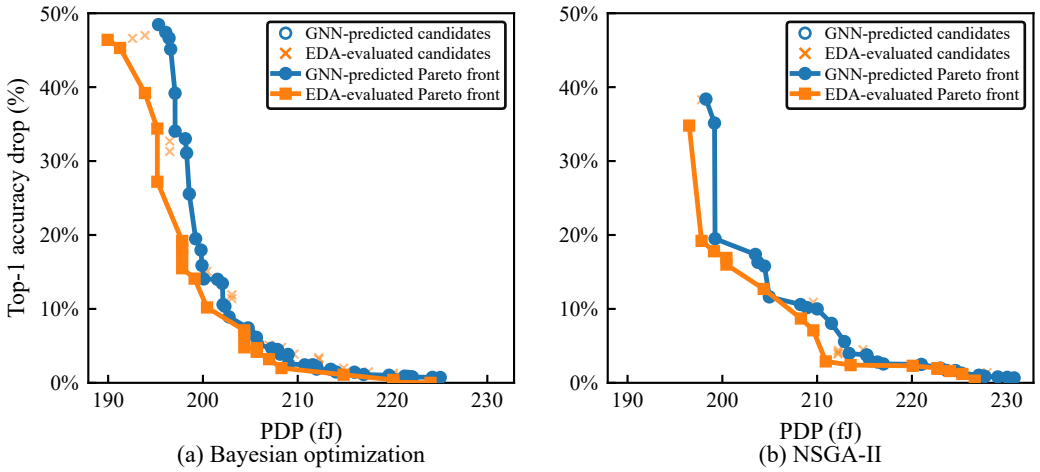


Figure 6: Bi-objective (Top-1 drop, PDP) fronts from the earlier 30,000-candidate benchmark, one panel per search algorithm. The area objective of Section 3.6 was not active, so these fronts are not directly comparable with those above. Faded markers are candidate designs and bold curves the Pareto fronts; blue is GNN-predicted and orange EDA-verified. The ordinate is the Top-1 drop in percent, and candidates above a 50 % drop are omitted.

E Extended Accuracy-Drop Prediction Results

Candidate pairs that conventional error metrics match closely yet whose measured Top-1 drops diverge are chosen after Test-label inspection and serve as post hoc illustrations, not as a pre-registered test. Figure 7 shows these pairs. Panel (a) plots the normalized mean error distance (NMED) of the

structural hold-out Test candidates against their measured Top-1 drop; panel (b) shows the two designs of each pair separately, sorted by their drop difference. Within a pair the NMED values differ by at most 4.4×10^{-5}, yet the measured drops differ by 68–82 percentage points, so two designs that a conventional error metric ranks as equivalent can sit at opposite ends of the accuracy scale.

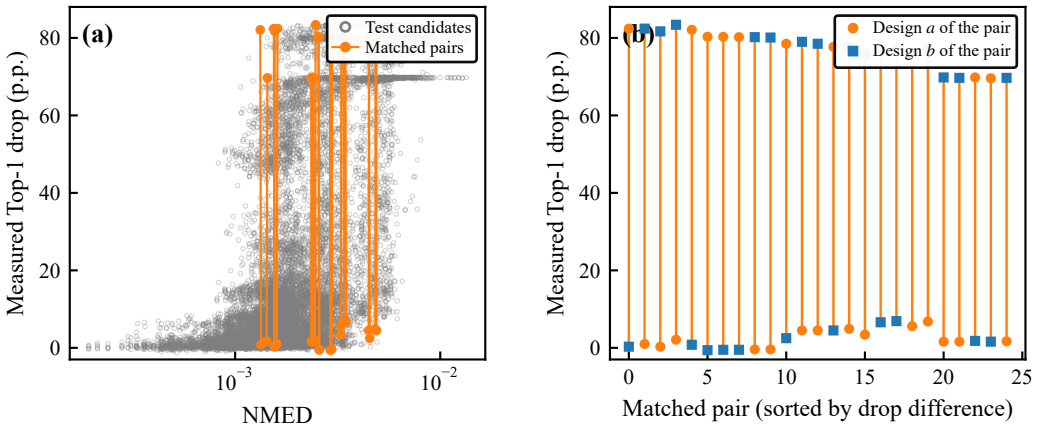


Figure 7: Candidates that conventional error metrics match closely yet whose measured Top-1 drops diverge. (a) NMED (log scale) against the measured Top-1 drop for the structural hold-out Test candidates; orange segments mark the matched pairs. (b) The same pairs sorted by drop difference, one marker per design: one design of every pair suffers a large drop while the other stays close to the exact baseline. Pairs are selected after Test-label inspection and serve as post hoc illustrations, not as a pre-registered test.