LLM-Guided Neural Architecture Search for Robust Co-Design of Physical Neural Networks

Tyler King [1 2] **Timothee Leleu** [1 3]

Abstract

Deploying neural networks on unconventional hardware demands architectures that co-optimize task accuracy and platform-specific constraints such as energy cost, physical non-idealities, and numerical precision. Existing neural architecture search (NAS) methods are typically tailored to a single hardware family, limiting cross-platform comparison and generalization. We introduce Unconventional Hardware Neural Architecture Search (UH-NAS), a hardware-agnostic, LLM-guided NAS framework that integrates language models as evolutionary operators to co-optimize accuracy and inference energy. By exposing hardware as a swappable backend with per-platform energy models, physical constraints, and non-ideality simulators, UH-NAS enables fair system-level comparisons across various backends without modifying the search algorithm. Tested on optical MZI hardware, UH-NAS discovers more diverse, robust architectures than conventional baselines while outperforming existing LLM-to-NAS approaches. Additional ablations on architecture robustness under non-idealities and the role of system prompts highlight the importance of architecture–hardware co-design for emerging computing platforms.

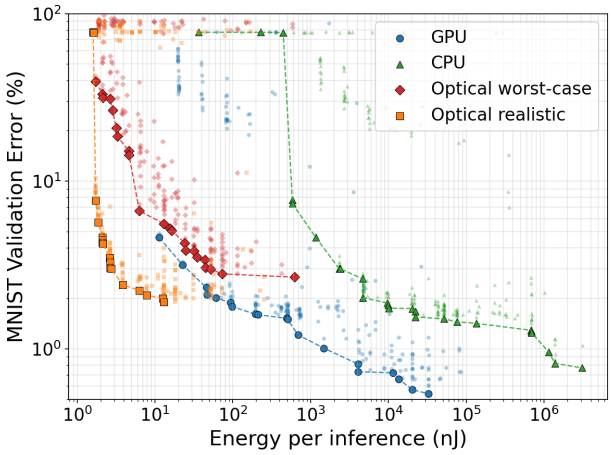

Figure 1. Pareto fronts of MNIST validation error (%) versus energy per inference (nJ) obtained with UH-NAS across multiple hardware backends, including GPU (Blackwell B200, FP16), CPU (Xeon 8380, FP32), and MZI mesh–based optical systems under both realistic and worst-case non-idealities. Optical hardware operates at substantially lower energy but with higher error rates due to device non-idealities; UH-NAS identifies robust architectures that narrow this gap.

1. Introduction

Neural architecture search (NAS) has become a standard framework for automating neural-network design, and a central lesson from hardware-aware NAS is that the best architecture depends on the target deployment platform rather than accuracy alone. Classical NAS work formalized the problem in terms of search space, search strategy, and performance estimation, while platform-aware methods showed that latency, energy, and device characteristics can change the optimal accuracy–cost trade-off (1; 2; 3; 4; 5; 6; 7; 8; 9). In other words, hardware does not merely rescale the score of a model, but may also change which architectural motifs are preferred.

This dependence on hardware is even stronger on unconventional hardware. In analog in-memory computing, non-ideal crossbars, and photonic tensor cores, performance is shaped not only by nominal throughput, but also by device non-idealities such as drift, noise, mismatch, crosstalk, and limited precision (10; 11; 12; 13; 14). Photonic computing provides a particularly compelling example: optical neural hardware promises high-throughput and energy-efficient linear operations, yet its behavior can be highly sensitive to phase errors, calibration, and physical design choices (15; 16; 17). Existing search methods for these settings are typically tailored to a single hardware family and rely on hand-designed objectives, search rules, or robustness mod-

[1]NTT Research [2]Princeton University [3]Stanford University. Correspondence to: Tyler King <tyler.king@ntt-research.com, tk9138@princeton.edu>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

els that may break down under slight hardware changes.

At the same time, recent work has shown that large language models (LLMs) can act as flexible search operators for architecture design, serving as mutation, reflection, and reasoning modules in evolutionary or quality-diversity NAS loops (18; 19; 20; 21; 22). This suggests a different route to hardware-aware NAS: rather than designing a new NAS algorithm for every emerging hardware, one can use an LLM to integrate hardware-specific context and adapt its search behavior through simulator feedback. Motivated by this idea, we introduce Unconventional Hardware Neural Architecture Search (UH-NAS), a robust LLM-guided NAS for unconventional hardware.

UH-NAS combines these two lines of work by integrating LLM-guided evolutionary search into a hardware-aware NAS framework for unconventional computing platforms. In contrast to prior NAS methods for unconventional hardware, which are often specialized to a single hardware paradigm with hand-designed objectives, UH-NAS models hardware through operation-level energy costs, physical constraints, and non-ideality models associated with standard neural-network operations, enabling the same search procedure to operate across heterogeneous platforms. Search is further guided by LLM-generated system prompts and a hardware-specific knowledge base updated after each NAS generation. Furthermore, unlike traditional NAS approaches that typically ignore hardware non-idealities and often rely on zero-cost (ZC) proxies (which we observe to break down under hardware non-idealities), we score every candidate via full noisy training, guided by the aforementioned hardware-specific design heuristics to help guide the search.

Our main results are as follows. First, the proposed framework enables consistent system-level comparison across heterogeneous hardware platforms (CPU, GPU, and MZI-based optical systems) by optimizing architectures separately for each hardware under its own constraints. Second, under strong hardware non-idealities, UH-NAS discovers more diverse architectures and achieves lower validation error than conventional NAS baselines. Third, architectures optimized under severe non-idealities differ qualitatively from those found in ideal settings, indicating that robustness cannot be extrapolated from ideal training. Finally, the method uncovers favorable accuracy–energy trade-offs on unconventional hardware such as optical systems, highlighting the importance of architecture–hardware co-design for revealing system-level advantages of emerging platforms. Analyzing such architectures yields physically interpretable design principles for unconventional hardware platforms, bridging the gap between NAS-discovered architectures and physical hardware understanding.

2. Related Work

2.1. NAS for Unconventional Computing

As unconventional computing platforms improve in both energy efficiency and accuracy, the need for machine-learning architectures capable of fully realizing the advantages of these emerging hardware systems continues to grow. Conventional NAS methods have been applied across a wide range of unconventional hardware paradigms to automate the discovery of architectures that fully exploit platform-specific strengths, notably in analog in-memory computing (9; 10; 12; 23), photonics (14; 24), quantum computing (13; 25; 26), and spiking/neuromorphic computing (27; 28).

Despite the promising performance of NAS for unconventional computing, existing approaches are often hardware-specific and rely on hand-designed objectives and heuristics, limiting their ability to adapt to varying hardware characteristics and non-idealities without hardware-specific expertise. Recent LLM-assisted search and algorithm-discovery methods provide a promising direction for overcoming these limitations by incorporating hardware-specific knowledge directly into the search process.

2.2. Evolutionary Algorithms and Algorithm Discovery

LLM-driven evolution has recently emerged as a powerful paradigm for autonomous algorithm discovery, with FunSearch (29), AlphaEvolve (30), and AlphaTensor (31) producing breakthrough results on long-standing mathematical and algorithmic problems, while EoH (32) and ReEvo (33) extend the paradigm to combinatorial heuristic design through refinement loops. Building on this line of work, we adopt NSGA-II (34) as our evolutionary backbone. It natively handles the multi-objective nature of NAS, allowing UH-NAS to construct high-performing accuracy–energy Pareto fronts for cross-hardware comparisons while also preserving population diversity.

2.3. LLM-assisted NAS

Recent advances in LLM-assisted algorithm discovery naturally extend to neural architecture search on conventional hardware platforms. These LLM-to-NAS approaches typically utilize LLMs to either predict components as part of evolutionary architectures (18; 35) or prompt them to propose novel architectures (22; 21; 20; 19). To the best of our knowledge, however, these LLM-guided NAS approaches have primarily been studied on conventional digital electronics (e.g., CPU/GPU), with comparatively limited exploration in unconventional hardware.

To accelerate their search, many such NAS algorithms leverage ZC (zero cost) proxies for architecture feedback (36; 22; 21), allowing for model evaluation at initialization.

Despite this promise, we observe that ZC proxies break down under hardware non-idealities, which are particularly prevalent in unconventional computing schemes (16; 12; 9).

Critically, unconventional hardware breaks key assumptions underlying existing LLM-to-NAS methods: zero-cost proxies, widely adopted to accelerate NAS search (36; 22; 21), exhibit negative correlations with validation accuracy under hardware non-idealities (Figure 5), hardware-specific constraints invalidate large portions of the search space (e.g. prohibitively expensive skip connections), and an architecture's clean performance is not predictive of its noisy performance (16; 12; 9). Simply integrating a new energy model into existing frameworks is insufficient to address such issues as it would fail to incorporate constrained search spaces, non-ideality-aware evaluation or hardware-specific knowledge. UH-NAS addresses such issues through its hardware-agnostic backend abstraction, full noisy training of every candidate, and LLM-synthesized hardware-specific design heuristics.

Algorithm 1 UH-NAS: Hardware-Adaptive LLM-Guided NAS

Input: Hardware backend $\mathcal{B}_i$, dataset $\mathcal{D}$, generations T, candidates per gen N.
Output: Pareto front $\mathcal{P}^*$ (accuracy $\uparrow$, energy $\downarrow$)

Initialize population A_0 from seed architectures.
$\mathcal{K} \leftarrow \mathcal{B}_i.\text{DesignHeuristics}()$ *// knowledge base*
for $t = 0, 1, \ldots, T$ **do**
  $\mathcal{K} \leftarrow \text{LLM-UpdateKB}(\mathcal{K},\ \text{ParetoFront}(A_t),\ \mathcal{B}_i)$
  for $k = 1, \ldots, N$ **do**
    $x' \leftarrow \text{LLM-Generate}(\mathcal{K},\ \text{Select}(A_t),\ \mathcal{B}_i)$
    Score x': $z \leftarrow \text{LLM-Score}(x')$, *// accuracy*
    $e \leftarrow \mathcal{B}_i.\text{Energy}(x')$.
  end for
  Keep top candidates by ($z \uparrow$, $e \downarrow$).
  for each surviving candidate x **do**
    $\text{acc} \leftarrow \text{Train}(x,\ \mathcal{B}_i,\ \mathcal{D})$
    $\text{deg} \leftarrow \text{acc} - \text{Eval}(\mathcal{B}_i.\text{InjectNoise}(x),\ \mathcal{D})$
    Insert $(x,\ \text{acc},\ \mathcal{B}_i.\text{Energy}(x),\ \text{deg})$ into A_{t+1} via NSGA-II
  end for
end for
return $\mathcal{P}^* \leftarrow \text{ParetoFront}(A_{t+1})$

3. Method

3.1. Problem Formulation

Our optimization goal is to find candidate architectures that generate a Pareto front of our proposed multi-objective optimization problem given a specific hardware type. For each hardware backend $i \in \{\text{optical}, \text{gpu}, \text{cpu}\}$, UH-NAS evaluates a population of candidate architectures $\{x_{i,k}\}_{k=1}^{N_i}$, where k indexes architectures searched for that backend. For notational simplicity, we write x_i when referring to a generic candidate architecture under hardware backend i. More formally, for a candidate architecture x_i evaluated on hardware backend i, we seek to compute $\min_{x_i} F_i(x_i)$, where

$$F_i(x_i) = \big(f_{i,1}(x_i),\ f_{i,2}(x_i),\ \ldots\big) \tag{1}$$

denotes a vector of hardware-dependent objective functions. From this, we construct a Pareto front, defined as the set of architectures x_i that are not strictly dominated in this objective space (i.e. for which no other candidate architecture achieves both higher accuracy and lower energy). More formally, the Pareto front $\mathcal{P}_i$ is defined as:

$$\begin{aligned}\mathcal{P}_i = \{F_i(x_i)\ :\ x_i \in \mathcal{X}_i,\ \nexists\, x_i' \in \mathcal{X}_i \\ \text{s.t. } F_i(x_i') \preceq F_i(x_i) \text{ and } F_i(x_i') \neq F_i(x_i)\}.\end{aligned} \tag{2}$$

Proof-of-concept instantiation setup For the proof-of-concept considered in this work, we restrict the optimization problem to maximizing classification accuracy $f_{i,\text{acc}}(x_i)$ and minimizing inference energy $f_{i,\text{eff}}(x_i)$. More formally,

$$F_i(x_i) = \big(-f_{i,\text{acc}}(x_i),\ f_{i,\text{eff}}(x_i)\big), \tag{3}$$

where the inference energy $f_{i,\text{eff}}(x_i)$ is approximated using platform-specific analytic cost models based on the structure of the model.

We instantiate the proposed framework on MNIST classification, where a candidate x_i is a PyTorch program defining a neural network. Given a candidate x_i, we train the model and evaluate the classification accuracy $f_{i,\text{acc}}(x_i; \mathcal{B}_i)$ on a validation set. The training procedure is mildly hardware-aware: depending on the hardware regime $\mathcal{B}_i$ (consisting of hardware non-idealities, constraints, energy model), certain layers are replaced by quantization-aware variants (e.g., 8-bit optical models), providing a first-order approximation of hardware effects. We additionally integrate hardware constraints (e.g., expensive skip connections in optical computing) and hardware non-idealities (e.g. phase error, crosstalk in optical computing). A full list of hardware constraints and non-idealities is provided in Section 3.3.

This implementation provides a minimal but functional realization of hardware-aware LLM-to-NAS, where architectures are jointly evaluated with respect to task loss and hardware-dependent cost.

3.2. UH-NAS Implementation

3.2.1. Overall Framework

UH-NAS serves as an iterative co-evolutionary framework that includes four stages per generation. Similar to prior NAS architectures (21; 22), we integrate LLM-in-the-loop, utilizing it both as an evolutionary operator and a knowledge synthesizer that iteratively refines NAS design heuristics.

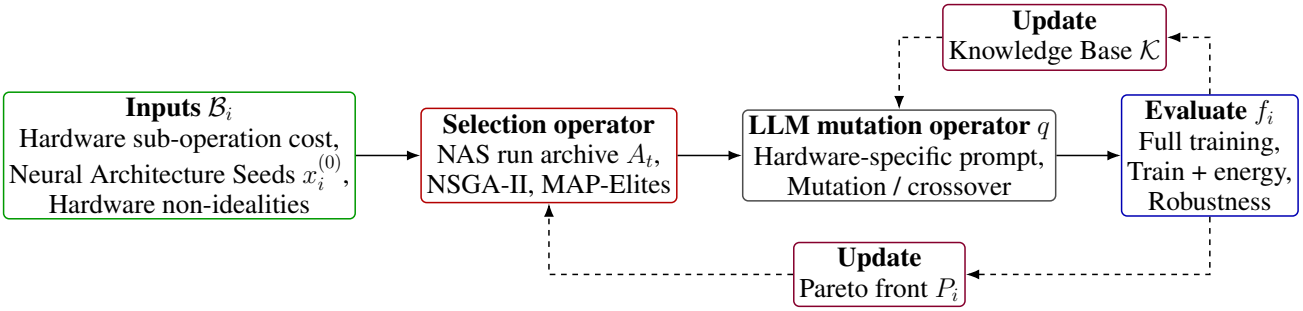

Figure 2. Overview of the UH-NAS framework. Each generation begins with hardware-specific inputs (sub-operation energy costs, seed architectures, and non-ideality parameters). A selection operator (NSGA-II with MAP-Elites diversity) chooses parents, which the LLM mutates or recombines via hardware-specific prompts. Candidates are trained and evaluated for accuracy, energy, and robustness under non-idealities. Resulting scores update the Pareto front and a knowledge base of design heuristics that guide subsequent LLM generations.

Central to our architecture, UH-NAS is hardware-agnostic: all hardware-specific information (energy cost models, physical constraints, non-idealities, and quantization) are incorporated as replaceable hardware backend abstractions, allowing for easy comparisons between different hardware implementations without modifying the underlying NAS algorithm. These hardware considerations (Section 3.3) allow for accurate estimation of model energy costs. We further integrate system prompts (imposing physical constraints found in certain hardware implementations), which we ablate in Table 4, along with following LLM-NAS's (22) knowledge base (updatable architecture-specific rules updated every NAS generation that guide its search).

The full pseudocode for UH-NAS is described in Algorithm 1.

3.2.2. Population Initialization

The search space begins by instantiating a diverse set of hand-designed architectures $x_i^{(0)}$ that cover different parts of the search space (MLP, convolutional-linear, etc.). The different layers are described as typed blocks within an architectural vocabulary, similar to RZ-NAS (21), and are concatenated to form a block string that is validated by a forward pass. This vocabulary is updated depending on the hardware being tested (and this is passed in as part of the LLM generation prompt).

3.2.3. Population Augmentation

At each NAS generation, the LLM receives the global Pareto front, the recently evaluated candidates, and the accuracy degradation induced by hardware non-idealities. From these, the LLM is prompted to update the design heuristics as a structured list, allowing the LLM to discover robustness patterns not inherently visible from model performance.

New candidate architectures are generated through mutation and crossover operations guided by this evolving knowledge base. Parents are selected from the current archive based on multi-objective fitness and diversity criteria, while the LLM proposes architectural modifications conditioned on prior search history. Formally, candidate generation follows

$$x_i^{(t+1)} \sim q(\cdot \mid A_t, \mathcal{K}). \tag{4}$$

We denote by $q(\cdot \mid A_t, \mathcal{K})$ the LLM-induced proposal distribution over candidate architectures, conditioned on the current archive A_t and the hardware-specific knowledge base $\mathcal{K}$. Then, for each candidate in a generation, a target region in a MAP-Elites diversity grid is sampled with a 70/30 exploit/explore ratio, similar to LLMatic (20). Parents are selected from the NSGA-II population via binary tournament (34). This is then used to prompt the LLM, encouraging either crossover or mutation based on its priors to generate new candidates. More information on our multi-objective optimization is found in Section 3.2.4.

For candidate architectures, we perform a full training loop consisting of 20-epoch training with Adam optimizer and cross-entropy loss, along with quantization-aware training (QAT) support for unconventional hardware with limited bit precision and non-ideality robustness calculations. The candidate is then inserted back into the NSGA-II population with information about its accuracy, energy, and robustness degradation.

3.2.4. Multi-Objective Selection

We leverage NSGA-II (34) for Pareto front generation, co-optimizing accuracy (↑) and inference energy (↓), although our framework supports arbitrary objective sets. Ranked Pareto fronts are generated, where rank-0 candidates are not dominated by any other individual, rank-1 candidates are dominated solely by rank-0 members, etc. Crowding distances are used to quantify the isolation of each candidate, encouraging spread across the front.

For parent selection, a binary tournament is used where two candidates are drawn at random, with the lower Pareto rank chosen as a parent. For crossover, two tournaments

produce two parents while for mutation, a single tournament suffices. After candidates are evaluated and inserted, the population is truncated, yielding the output: a final rank-0 Pareto front representing the set of non-dominated accuracy/energy tradeoffs.

Given the poor correlation between ZC methods and non-ideality accuracy observed in Figure 5, we decide to train all architectures from scratch (as in LLMatic (20)). We posit this slight negative correlation between non-ideal accuracy and ZC proxies can lead to destructive performance in NAS for MZI-based optical neural networks (ONNs).

3.3. Hardware Considerations

We model each hardware backend through an energy cost function and, for the optical regime, an additional structured non-ideality model. Table 1 summarizes per-operation energy estimates across CPU (Xeon 8380, FP32), GPU (B200, FP16), and the MZI-based ONN (8-bit precision). Linear MAC costs span roughly four orders of magnitude and are significantly cheaper on optical hardware, with the tradeoff being additional DAC/ADC conversion costs at network inputs and outputs that partially offset per-MAC advantage. Exact derivations for each are included in Appendix A.1.

	CPU (Xeon 8380)	**GPU** (B200)	**Optical** (MZI)
Linear MAC	91.7 pJ	0.89 pJ	0.02 pJ
Nonlinear (ReLU)	3 pJ	3 pJ	10 pJ
DAC (input)	—	—	2 pJ
ADC (output)	—	—	4 pJ
Precision (bits)	32 (FP32)	16 (FP16)	8

Table 1. Per-operation analytical energy proxy coefficients for CPU, GPU, and optical (MZI ONN) inference. Linear MAC is per multiply-accumulate; nonlinear, DAC, and ADC are per element. DAC/ADC are not applicable to digital CPU/GPU inference.

For the optical backend, we further capture three physical noise sources: phase error (σ_ϕ), thermal crosstalk (ε), and gamma noise (σ_γ) in Table 2. Realistic and worst-case noise values correspond to approximate lower and upper bounds reported in silicon-photonic literature, while quantization noise is handled separately via QAT for 8-bit representations that are commonplace in ONN literature. Full derivations of the energy model, the architectural constraints imposed on MZI meshes (block-diagonal decomposition, width caps, residual-connection penalties), and detailed descriptions of each noise source are provided in Appendix A.2.

	Phase error (σ_ϕ)	**Crosstalk** (ε)	**Gamma noise** (σ_γ)
Realistic	0.01	0.01	0.001
Medium	0.03	0.08	0.003
Worst	0.05	0.15	0.005

Table 2. Hardware non-ideality levels evaluated for optical regime. Realistic-case and worst-case values correspond to the lower and upper bounds reported in literature; medium-case is the midpoint.

4. Experiments

4.1. Experimental Setup

We evaluate UH-NAS on three hardware backends: a silicon-photonics MZI-mesh backend (simulated via `pytorch-onn`), a GPU reference backend (Blackwell on FP16), and a CPU reference backend (Xeon on FP32) tested on MNIST. The search procedure runs for 30 generations with 8 candidate architectures generated per generation. The NSGA-II population size is set to 20. Each candidate is trained for 20 epochs using Adam (lr=10^{-3}) with cross-entropy loss. Seed populations consist of 10 hand-designed architectures spanning pure MLP, convolutional-linear hybrid, and varying-depth configurations.

For competing architectures (LLMatic, RZ-NAS), we leverage the same seed architectures and optimize for non-ideality accuracy to encourage fairer comparisons, along with updating the search space to utilize the same MZI-typed vocabulary. For LLMatic, we utilize the same training procedure (30 generations, 8 candidate architectures per generation trained under the same conditions), maintaining the MAP-Elites archive and algorithm (same mutation and crossover prompts, archive sizes, curiosity/temperature update rules, etc.), switching the fitness signal to non-ideality accuracy for non-ideal LLMatic. For RZ-NAS we evaluate 1500 generations with a population size of 100 and utilize SynFlow as the zero-cost fitness function, also reporting non-ideality accuracy on the final-trained winner. We apply no additional constraints on parameter count/model size, but still utilize QAT for training and implement the same MZI ONN architectural limitations.

4.2. Cross-Platform Pareto Analysis: Summary

Figure 1 summarizes UH-NAS search outcomes across GPU, CPU, and MZI-based optical backends through their respective Pareto-optimal accuracy–energy trade-offs. A key advantage of the framework is enabling system-level, fair comparison across hardware platforms, as architectures are optimized independently for each hardware under its specific constraints and non-idealities. GPU architectures achieve significantly lower energy per inference than CPU architectures at comparable validation error. In contrast,

Table 3. Top-1 MNIST accuracy under worst-case optical non-idealities for UH-NAS and baselines (RZ-NAS, LLMatic), each using three LLM backbones. All methods share the same seed architectures and optimize for non-ideality accuracy. UH-NAS with GPT-4.1 achieves the highest mean accuracy (97.00%), outperforming both baselines and the ablation without LLM-guided generation.

Method	GPT-4.1 Nano	GPT-4.1	Gemini 3.1 Flash-Lite
UH-NAS	91.96% ± 0.43%	**97.00% ± 0.30%**	**96.95% ± 0.03%**
UH-NAS (w/o LLM)	92.74% ± 0.18%	92.46% ± 0.35%	92.61% ± 0.31%
RZ-NAS (21)	46.42% ± 20.71%	88.69% ± 6.85%	19.16% ± 8.11%
LLMatic (20)	47.07% ± 38.00%	66.03% ± 14.51%	51.93% ± 23.70%
LLMatic (non-ideal)	**93.40% ± 1.45** %	91.81% ± 1.83%	92.92% ± 2.09%

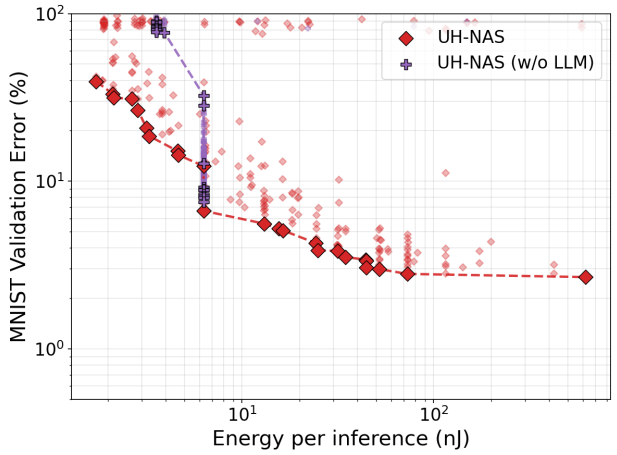

Figure 3. Ablation comparing UH-NAS (with LLM) and UH-NAS without LLM on the MZI optical backend under worst-case non-idealities. Without LLM-guided generation, the search degenerates to minor seed architecture modifications, producing only 26 unique designs out of 250 evaluations and a restricted Pareto front. With the LLM, UH-NAS explores 203 unique architectures, yielding broader coverage and lower validation error at comparable energy budgets.

optical hardware operates in a substantially lower energy regime, with a moderate accuracy degradation under non-idealities. UH-NAS reduces this gap by identifying architectures that improve robustness, indicating that optical platforms can approach competitive accuracy while retaining strong energy-efficiency advantages.

4.3. Benefits of LLM-Guided Search for Hardware-Aware NAS

In Table 3, compared to RZ-NAS and LLMatic, we note that UH-NAS tends to obtain higher top accuracy on MNIST (Table 3), along with being capable of co-optimizing both accuracy and efficiency (while other algorithms focus solely on performance thus leading to very deep searched architectures). We posit this advantage stems from UH-NAS's hardware awareness, driven in part by the LLM prompt guiding the algorithm. We observe that UH-NAS with a strong LLM backbone noticeably outperforms existing baselines, even when these baselines are explicitly trained/optimized for non-ideal accuracy (e.g. LLMatic with non-ideality accuracy as a criterion). This LLM backbone plays a key role in the system prompt/knowledge base generation, as weaker models such as GPT-4.1-nano tend to introduce weak or incorrect prompts that degrade search performance, while stronger models leverage the structured context to achieve large gains (Appendix C). We note that the poor performance of Gemini 3.1 Flash-Lite without system prompts is induced by the fact that it tends to bias towards convolutional layers, which exhibit catastrophic performance degradation under non-ideality conditions.

Figure 3 shows that UH-NAS without the guiding LLM rapidly collapses to a narrow region of the search space, yielding limited architectural diversity (26 valid unique architectures out of 250 evaluations) and a restricted Pareto front. In contrast, UH-NAS explores a substantially broader design space (203 valid unique architectures), resulting in improved coverage and lower validation error at comparable energy. This indicates that LLM-guided generation enables non-local exploration and mitigates search degeneracy induced by hardware constraints.

4.4. Robust Architecture Discovery Under Non-Idealities

Figure 4 evaluates the robustness of architectures discovered under worst-case non-idealities by re-evaluating the same Pareto front across varying noise regimes. Performance degrades with increasing non-idealities, but the Pareto front remains well-structured, indicating that architectures discovered under worst-case conditions retain strong performance across all regimes (while the inverse is not necessarily true). Notably, the worst-case-trained architectures remain competitive even under moderate and realistic non-idealities, suggesting that robustness learned during search generalizes beyond the specific noise level used for optimization.

Figure 5 reports Spearman correlations on 200 NAS-Bench-201 architectures under both clean and worst-case MZI ONN conditions. Under clean conditions, we observe correlations in line with prior ZC NAS literature (21; 36; 22), but under non-ideality conditions nearly every proxy exhibits negative correlations, mostly induced by the destructive performance of convolutional layers under worst-case

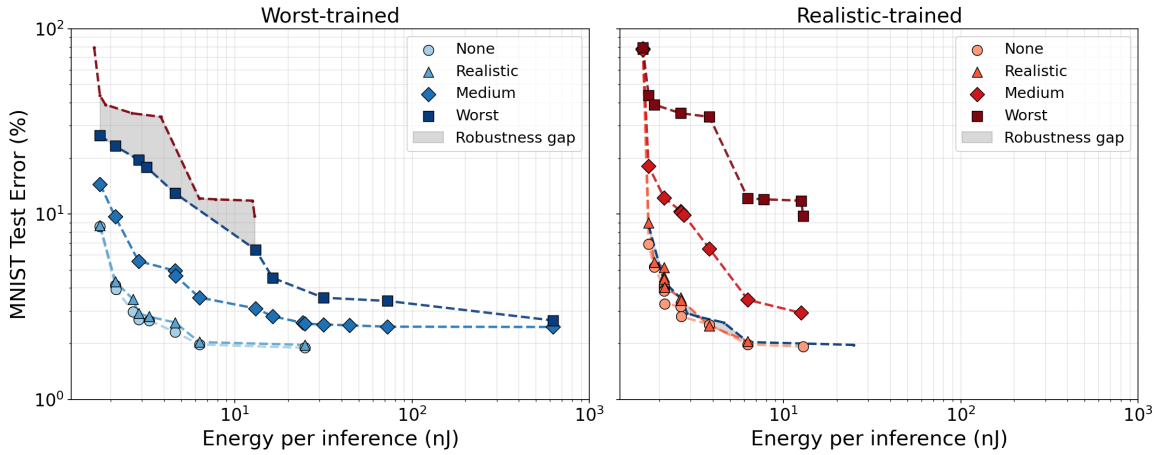

Figure 4. Robustness of UH-NAS–discovered architectures across non-ideality regimes on the MZI optical backend. The Pareto front optimized under worst-case noise (left) and realistic noise (right) is re-evaluated at four non-ideality levels: none, realistic, medium, and worst-case, with the shaded gray region highlighting the robustness gap of the other training environment (worst-case comparison on the left, realistic on the right). In both panels, the Pareto front structure is preserved across regimes, with a consistent upward shift in error as noise increases. Worst-case–trained architectures remain competitive even under moderate and realistic conditions, suggesting that robustness generalizes beyond the specific noise level used during search.

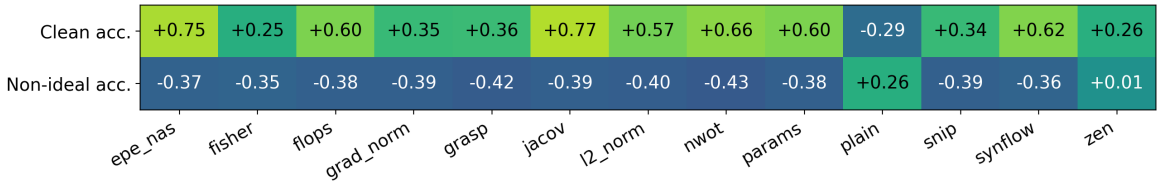

Figure 5. Spearman rank correlation between 13 zero-cost (ZC) proxies and validation accuracy on 200 randomly drawn NAS-Bench-201/CIFAR-10 architectures, evaluated under clean and worst-case MZI ONN non-idealities ($\sigma_\phi = 0.05$, $\sigma_\gamma = 0.005$, crosstalk$= 0.15$). Under clean conditions, most proxies show positive correlations consistent with prior literature. Under worst-case non-idealities, nearly all proxies exhibit negative correlations, indicating that ZC proxies are unreliable for NAS on non-ideal optical hardware.

non-idealities (which typically exhibit strong positive correlations with ZC methods). These findings justify UH-NAS's design decision to score every candidate via full training, while explaining the poor performance of ZC proxy NAS methods observed in Table 3.

4.5. Ablation Studies

Table 4 compares UH-NAS with and without the hardware-informed knowledge base ($\mathcal{K}$) under worst-case non-idealities. The results show that incorporating structured context about the hardware improves performance for stronger LLMs (e.g., GPT-4.1 and Gemini 3.1 Flash-Lite), highlighting the importance of hardware-aware prompting and contextual guidance during search.

Figure 6 compares the evolution of best accuracy across generations for different LLMs. Stronger models (GPT-4.1 and Gemini 3.1 Flash-Lite) achieve both faster convergence and higher final accuracy compared to smaller models (GPT-4.1 nano), which shows little to no improvement over iterations. This indicates that LLM capability directly impacts the effectiveness of the search process, with more capable models enabling non-trivial exploration and more efficient identification of high-performing architectures.

4.6. Discovered Architectures

On both CPU and GPU backends, we observe that high-accuracy Pareto-optimal architectures tend to follow traditional machine learning design principles, operating as convolutional-linear hybrid patterns with 4-6 layers (akin to LeNet-5 (37)) that exploit depth to maximize representational capacity. However, on the optical backend we observe a different trend: shallow, wide MLPs tend to dominate, with convolutional layers largely absent outside of the energy-efficient end of the Pareto front. In all settings, normalization frequently appears at high-energy regimes, and is particularly helpful for settings with worst-case non-idealities. We observe no skip connections in the optical Pareto front, likely due to the large overhead costs associated with signal re-routing and additional ADC/DAC conversions.

Such hardware differences are physically grounded. Depth

Table 4. Ablation on the role of the hardware-informed system prompt ($\mathcal{K}$) in UH-NAS under worst-case optical non-idealities. Removing the system prompt reduces accuracy across strong LLM backbones (GPT-4.1, Gemini 3.1 Flash-Lite), indicating that structured hardware context is essential for effective LLM-guided search given accurate construction of the prompt.

Method	GPT-4.1 Nano	GPT-4.1	Gemini 3.1 Flash-Lite
UH-NAS	91.96% ± 0.43%	**97.00% ± 0.30%**	**96.95% ± 0.03%**
UH-NAS (w/o $\mathcal{K}$)	**93.21% ± 2.34%**	96.02% ± 1.59%	96.55% ± 0.42%

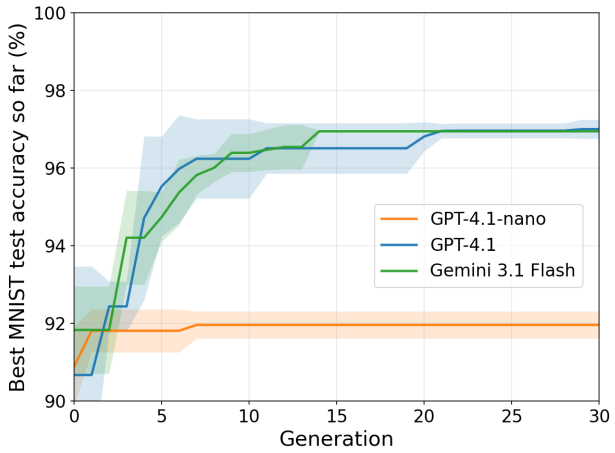

Figure 6. Best observed MNIST accuracy (worst-case non-idealities) over 30 NAS generations, averaged over 3 runs, for three LLM backbones. GPT-4.1 and Gemini 3.1 Flash-Lite converge to approximately 97% accuracy within 10–15 generations, while GPT-4.1-nano shows limited improvement due to its reduced capacity for generating effective system prompts and diverse architectures.

sensitivity arises due to phase error (σ_ϕ) compounding through sequential MZI mesh layers. Convolutional layers fail catastrophically due to crosstalk, which propagates weight-shifted convolutional kernel errors across the output feature map (38). Batch normalization, by comparison, operates as a noise recalibration mechanism, re-centering activations after noisy operations introduced by phase errors and crosstalk. Finally, the lack of skip connections and nonlinearities occur due to the extreme overhead from ADC/DAC costs, which are ≥ 2 orders of magnitude more expensive than optical MACs.

These findings provide a set of actionable design principles for MZI-based ONN that can help guide future architectures: namely that shallow, wide networks are preferred, utilizing solely linear layers under high crosstalk and integrating batch normalization to mitigate non-ideality model drift. These principles highlight a key feature of UH-NAS: the discovery of novel, robust architectures across heterogeneous hardware platforms. Compared to traditional NAS methods, our approach discovers a multitude of optimal architectures at varying energy budgets, avoiding common pitfalls such as overweighting deep architectures that may not be as resistant to hardware non-idealities or the utilization of ZC proxies that incorrectly score highest performers.

5. Conclusion

We presented UH-NAS, an LLM-guided NAS framework for unconventional hardware that co-optimizes accuracy and inference energy across heterogeneous platforms via swappable backend abstractions, enabling fair system-level comparisons. We integrate LLM feedback to guide the NAS, constraining the search space and iteratively delivering hardware-aware feedback. On MZI-based optical hardware under worst-case non-idealities, UH-NAS achieves 97% MNIST accuracy while discovering more diverse architectures compared to conventional baselines, with Pareto fronts that remain well-structured across noise regimes.

Beyond framework performance, UH-NAS uncovers physically interpretable design principles for ONNs: shallow, wide MLPs outperform convolutional-linear hybrids due to multiplicative phase noise compounding through depth and correlated crosstalk inducing errors that occur in convolutional weight reuse. Furthermore, batch normalization acts as a noise recalibration mechanism and skip connections are absent due to prohibitive ADC/DAC conversion costs. We hope these principles guide future neural networks deployed on photonic accelerators, bridging the gap between automated search and physical hardware.

Several limitations remain, however. UH-NAS assumes feedforward architectures composed of standard layers, which may not extend to fundamentally different hardware such as spiking neuromorphic processors or quantum computing. Additionally, while the framework is algorithmically hardware-agnostic, it still requires hardware-specific inputs, including seed architectures, energy models, and non-ideality simulators. Natural extensions include scaling to harder datasets, expanding to additional unconventional computing paradigms (e.g. memristors), and reducing computational costs through early-stopping strategies such as Hyperband-style successive halving (39).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

[1] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[2] 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.

[3] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan Celine Lin. Hw-nas-bench: Hardware-aware neural architecture search benchmark. *arXiv preprint arXiv:2103.10584*, 2021.

[4] Guihong Li, Sumit K Mandal, Umit Y Ogras, and Radu Marculescu. Flash: F ast neura l a rchitecture s earch with h ardware optimization. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(5s):1–26, 2021.

[5] Weiwen Jiang, Qiuwen Lou, Zheyu Yan, Lei Yang, Jingtong Hu, Xiaobo Sharon Hu, and Yiyu Shi. Device-circuit-architecture co-exploration for computing-in-memory neural accelerators. *IEEE Transactions on Computers*, 70(4):595–605, 2020.

[6] Zhihang Yuan, Jingze Liu, Xingchen Li, Longhao Yan, Haoxiang Chen, Bingzhe Wu, Yuchao Yang, and Guangyu Sun. Nas4rram: neural network architecture search for inference on rram-based accelerators. *Science China Information Sciences*, 64(6):160407, 2021.

[7] Chuteng Zhou, Fernando Garcia Redondo, Julian Büchel, Irem Boybat, Xavier Timoneda Comas, SR Nandakumar, Shidhartha Das, Abu Sebastian, Manuel Le Gallo, and Paul N Whatmough. Analognets: Ml-hw co-design of noise-robust tinyml models and always-on analog compute-in-memory accelerator. *arXiv preprint arXiv:2111.06503*, 2021.

[8] Shubham Negi, Indranil Chakraborty, Aayush Ankit, and Kaushik Roy. Nax: neural architecture and memristive xbar based accelerator co-design. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 451–456, 2022.

[9] Aniss Bessalah, Hatem Mohamed Abdelmoumen, Karima Benatchba, and Hadjer Benmeziane. Analognas-bench: A nas benchmark for analog in-memory computing. *arXiv preprint arXiv:2506.18495*, 2025.

[10] Hadjer Benmeziane, Corey Lammie, Irem Boybat, Malte Rasch, Manuel Le Gallo, Hsinyu Tsai, Ramachandran Muralidhar, Smail Niar, Ouarnoughi Hamza, Vijay Narayanan, et al. Analognas: A neural network design framework for accurate inference with analog in-memory computing. In *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, pages 233–244. IEEE, 2023.

[11] Zheyu Yan, Da-Cheng Juan, Xiaobo Sharon Hu, and Yiyu Shi. Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 859–864, 2021.

[12] Abhiroop Bhattacharjee, Abhishek Moitra, and Priyadarshini Panda. Xplorenas: Explore adversarially robust and hardware-efficient neural architectures for non-ideal xbars. *ACM Transactions on Embedded Computing Systems*, 22(4):1–17, 2023.

[13] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 692–708. IEEE, 2022.

[14] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Mingjie Liu, Shuhan Zhang, Ray T Chen, and David Z Pan. Adept: Automatic differentiable design of photonic tensor cores. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 937–942, 2022.

[15] Yichen Shen, Nicholas C. Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, and Marin Soljačić. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7):441–446, 2017.

[16] Ryan Hamerly, Liane Bernstein, Alexander Sludds, Marin Soljačić, and Dirk Englund. Large-scale optical neural networks based on photoelectric multiplication. *Physical Review X*, 9(2), 2019.

[17] Bhavin J Shastri, Alexander N Tait, Thomas Ferreira de Lima, Wolfram HP Pernice, Harish Bhaskaran, C David Wright, and Paul R Prucnal. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics*, 15(2):102–114, 2021.

[18] Angelica Chen, David Dohan, and David So. Evoprompting: Language models for code-level neural architecture search. *Advances in neural information processing systems*, 36:7787–7817, 2023.

[19] Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. Can gpt-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*, 2023.

[20] Muhammad Umair Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher Cleghorn. Llmatic: Neural architecture search via large language models and quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '24, page 1110–1118. ACM, 2024.

[21] Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. RZ-NAS: Enhancing LLM-guided neural architecture search via reflective zero-cost strategy. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 27237–27254. PMLR, 13–19 Jul 2025.

[22] Hengyi Zhu, Grace Li Zhang, and Shaoyi Huang. Llm-nas: Llm-driven hardware-aware neural architecture search, 2025.

[23] Olga Krestinskaya, Mohammed E. Fouda, Hadjer Benmeziane, Kaoutar El Maghraoui, Abu Sebastian, Wei D. Lu, Mario Lanza, Hai Li, Fadi Kurdahi, Suhaib A. Fahmy, Ahmed Eltawil, and Khaled N. Salama. Neural architecture search for in-memory computing-based deep learning accelerators. *Nature Reviews Electrical Engineering*, 1(6):374–390, May 2024.

[24] Amin Shafiee, Febin Sunny, Sudeep Pasricha, and Mahdi Nikdast. Luxnas: A coherent photonic neural network powered by neural architecture search. In *CLEO 2025*. Optica Publishing Group, 2025.

[25] Wenjie Wu, Ge Yan, Xudong Lu, Kaisen Pan, and Junchi Yan. QuantumDARTS: Differentiable quantum architecture search for variational quantum algorithms. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 37745–37764. PMLR, 23–29 Jul 2023.

[26] Darya Martyniuk, Johannes Jung, and Adrian Paschke. Quantum architecture search: A survey. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, page 1695–1706. IEEE, 2024.

[27] Byunggook Na, Jisoo Mok, Seongsik Park, Dongjin Lee, Hyeokjun Choe, and Sungroh Yoon. AutoSNN: Towards energy-efficient spiking neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16253–16269. PMLR, 17–23 Jul 2022.

[28] Jiaqi Yan, Qianhui Liu, Malu Zhang, Lang Feng, De Ma, Haizhou Li, and Gang Pan. Efficient spiking neural network design via neural architecture search. *Neural Networks*, 173:106172, May 2024.

[29] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[30] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[31] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

[32] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

[33] Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[35] Caiyang Yu, Xianggen Liu, Yifan Wang, Yun Liu, Wentao Feng, Xiong Deng, Chenwei Tang, and Jiancheng Lv. Gpt-nas: Neural architecture search meets generative pre-trained transformer model. *Big Data Mining and Analytics*, 8(1):45–64, February 2025.

[36] Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. NAS-bench-suite-zero: Accelerating research on zero cost proxies. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Ye Su, Pengju Fu, Yichen Ye, Junxiong Chai, Xiao Jiang, Hongyu Yang, and Yiyuan Xie. Reliability analysis of optical neural networks with non-ideal signal transmission. *Optical Fiber Technology*, 87:103928, October 2024.

[39] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.

[40] NVIDIA Corporation. NVIDIA Blackwell Architecture Technical Brief. NVIDIA, 2024.

[41] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Efficient processing of deep neural networks: A tutorial and survey, 2017.

[42] Intel Corporation. Intel xeon platinum 8380 processor (60m cache, 2.30 ghz) — product specifications. Intel ARK (Product Information), SKU 212287, 2021. Accessed: 2026-04-24.

[43] Mitchell A. Nahmias, Thomas Ferreira de Lima, Alexander N. Tait, Hsuan-Tung Peng, Bhavin J. Shastri, and Paul R. Prucnal. Photonic multiply-accumulate operations for neural networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(1):1–18, January 2020.

[44] Ryan Hamerly, Saumil Bandyopadhyay, and Dirk Englund. Asymptotically fault-tolerant programmable photonics. *Nature Communications*, 13(1), November 2022.

[45] Mohamed Bouchakour, Emmanuel Bergeret, Gilles Sicard, Kamel Abdelouahab, and François Berry. A low power fully analog mac operation using standard cmos technology for neural network inference. In *2025 23rd IEEE Interregional NEWCAS Conference (NEWCAS)*, page 10–14. IEEE, 2025.

[46] Boris Murmann. ADC Performance Survey 1997-2026. [Online]. Available: https://github.com/bmurmann/ADC-survey.

[47] Ian A. D. Williamson, Tyler W. Hughes, Momchil Minkov, Ben Bartlett, Sunil Pai, and Shanhui Fan. Reprogrammable electro-optic nonlinear activation functions for optical neural networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(1):1–12, January 2020.

[48] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Ray T. Chen, and David Z. Pan. L2ight: Enabling on-chip learning for optical neural networks via efficient in-situ subspace optimization. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[49] Cansu Demirkiran, Furkan Eris, Gongyu Wang, Jonathan Elmhurst, Nick Moore, Nicholas C. Harris, Ayon Basumallik, Vijay Janapa Reddi, Ajay Joshi, and Darius Bunandar. An electro-photonic system for accelerating deep neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, 19(4):1–31, 2023.

[50] Sri Krishna Vadlamani, Dirk Englund, and Ryan Hamerly. Transferable learning on analog hardware. *Science Advances*, 9(28), 2023.

[51] Sunil Pai, Zhanghao Sun, Tyler W. Hughes, Taewon Park, Ben Bartlett, Ian A. D. Williamson, Momchil Minkov, Maziyar Milanizadeh, Nathnael Abebe, Francesco Morichetti, Andrea Melloni, Shanhui Fan, Olav Solgaard, and David A. B. Miller. Experimentally realized in situ backpropagation for deep learning in photonic neural networks. *Science*, 380(6643):398–404, April 2023.

[52] Maziyar Milanizadeh, Douglas Aguiar, Andrea Melloni, and Francesco Morichetti. Canceling thermal cross-talk effects in photonic integrated circuits. *Journal of Lightwave Technology*, 37(4):1325–1332, February 2019.

[53] Jiaqi Gu, Zheng Zhao, Chenghao Feng, Hanqing Zhu, Ray T. Chen, and David Z. Pan. Roq: A noise-aware quantization scheme towards robust optical neural networks with low-bit controls. In *2020 Design, Automation; Test in Europe Conference; Exhibition (DATE)*, page 1586–1589. IEEE, March 2020.

A. Hardware Modeling Costs

A.1. Energy Efficiency

For GPU inference, we estimate per-inference energy from an NVIDIA B200 Blackwell GPU operating at ideal FP16 tensor core utilization. Assuming a thermal design power of 1000 W and a peak throughput of 2.25×10^{15} FLOPS, each FP16 multiply-accumulate dissipates approximately $\frac{2\times 1000\ \mathrm{W}}{2.25\times 10^{15}\ \mathrm{FLOPS}} \approx 0.89$ pJ/MAC (40). The same calculation under FP8 doubles the throughput to 4.5×10^{15} FLOPS, yielding a more aggressive ~ 0.44 pJ/MAC; we use the FP16 figure throughout as it more closely matches the precision regime in our experiments. Element-wise nonlinear activations are estimated at 3 pJ/element, consistent with prior energy accounting for digital accelerators (41).

For CPU inference, we estimate per-inference energy from an Intel Xeon Platinum 8380 running single-threaded FP32 inference. Peak FP32 throughput is derived as 40 cores $\times$ 2 FMA units $\times$ 16 lanes $\times$ 2 ops $\times$ 2.3 GHz $\approx$ 5.89 TFLOPS, which combined with a 270 W TDP yields an FP32 MAC cost of $\frac{2\times 270\ \mathrm{W}}{5.89\times 10^{12}\ \mathrm{FLOPS}} \approx 91.7$ pJ/MAC (42). As with the GPU, we charge 3 pJ per element for ReLU and other element-wise nonlinearities (41). The roughly two-orders-of-magnitude gap between CPU and GPU MAC energy reflects the absence of dedicated tensor units and the lower arithmetic intensity of general-purpose CPU pipelines.

For the MZI-based ONN, we leverage a simple estimator that analytically computes per-inference energy based on the network's layer structure. Linear and convolutional operations executed on the MZI mesh are charged at 20 fJ/MAC, an estimate consistent with reported energy figures for silicon-photonic tensor cores (43; 16; 44). Because MZI meshes are inherently linear, every signal entering or leaving the optical domain incurs a data conversion cost: 2 pJ/element for input digital-to-analog conversion (45) and 4 pJ/element for output analog-to-digital conversion (46). Element-wise nonlinearities are performed in the electronic domain at 10 pJ/element (47), a substantially higher per-element cost than on digital hardware that reflects the overhead of opto-electronic conversion at each nonlinearity. The dominant per-MAC advantage of the optical regime is therefore partially offset by these fixed conversion costs, which become amortized for deeper or wider networks.

A.2. Non-ideality Model

For tested hardware, we consider multiple relevant factors, including quantization, hardware constraints, and hardware non-idealities in an attempt to closely simulate expected hardware performance. In the case of CPU and GPU, we note that there are no non-idealities that need to be considered.

For optical hardware, we simulate the performance of MZI meshes for deep learning as seen in Shen et al. (15), leveraging pytorch-onn (48) to implement MZI-based linear and convolutional layers. To accurately represent bit precision in photonic MZI meshes, we simulate 8-bit precision for MZI phase values and input/output signals (15; 49), training with quantization-aware training (QAT).

We additionally impose architectural constraints pertinent to MZI meshes. These include block-diagonal decomposition, decomposing weight matrices into 4×4 sub-matrices implemented as a Clements-topology MZI mesh (50). We also constrain the width of linear layers to ≤ 512 (a soft physical constraint) based on the maximally demonstrated MZI meshes (44), along with heavily penalizing residual connections because these require additional ADC/DAC conversions to allow signal summation. Due to the lack of nonlinear activations in optical computing, we require opto-electronic conversions that induce higher energy costs. Finally, we include ADC/DAC costs at networks input/output layers, resulting in negligible energy overhead for deeper networks.

For hardware non-idealities, we capture four relevant physical noise sources that typically arise in silicon-photonic chips (and that are provided through pytorch-onn's open-source framework). These include phase error, crosstalk, gamma noise, and quantization noise (the last of which is handled by our existing QAT scheme). Phase error represents Gaussian perturbations on MZI phase shifter values arising from fabrication imprecision, where $\sigma_\phi \in [0.01, 0.05]$ is a reasonable estimate for MZI ONNs (15; 51). We assume thermal crosstalk of $\varepsilon \in [0.01, 0.15]$ (modeled as nearest-neighbor weight-space leakage), which is consistent with estimates for silicon-photonic MZI meshes (44; 52). Finally, for gamma noise, we represent it as $\sigma_\gamma \in [0.001, 0.005]$, which captures a multiplicative per MZI voltage-to-phase conversion error (53). These non-idealities are injected by perturbing trained weight tensors in weight-space. These outputs are then evaluated on the validation set to produce non-ideal accuracy metrics, which are then fed into the LLM's knowledge base for future generations.

B. Energy Estimates and Validation

All hardware backends in this work currently compute per-inference energy from a closed-form analytical model described in Appendix A.1. These estimations are essential to the NSGA-II loop, as the evaluated candidates are sorted by both energy efficiency as well as accuracy. Although our numbers are pulled directly from product specifications, the question of how accurate these estimates are remains. To test this, we validate energy results by re-running every architecture along the CPU Pareto frontier on bare-metal

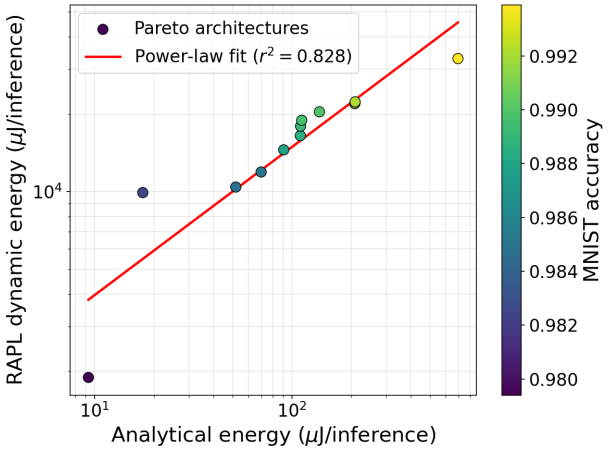

Figure 7. Log-log correlation between analytical energy estimates and real RAPL energy validation on an AWS `c6i.metal` instance with Xeon Platinum 8375C (Ice Lake-SP), 2.90 GHz, single core. Only models along the Pareto frontier were tested, giving a log-space correlation of $r^2 = 0.828$. Averaged over 50,000 forward passes.

Xeon silicon AWS `c6i.metal` instances, utilizing Intel's RAPL `MSR_PKG_ENERGY_STATUS` counter averaged over $N = 50{,}000$ batch-size-1 passes. Because AWS does not expose B200 instances at the time of writing, only CPU energy estimates are included.

Figure 7 and Table 5 compare the results between the closed-form analytical model and the real energy results. Although the analytical estimates under-estimate by 2 orders of magnitude on average, we note that the ranking fidelity is high: when using a least-squares fit, we obtain an r^2 of 0.828.

This offset is expected because the analytical model assumes best-case performance, and thus serves as a theoretical maximum bound. Because PyTorch inference induces additional computational overhead, the RAPL energy costs tend to be much higher. However, we note that every hardware backend tested in UH-NAS utilizes the same theoretical maximum convention, and thus cross-platform comparisons are still valid. It is worth noting then that architecture rankings by Pareto frontiers are best served as theoretical maximum comparisons, not as deployment-time comparisons.

C. System Prompt Analysis

UH-NAS leverages hardware-specific system prompts to condition the LLM on physical constraints, non-idealities, and energy-aware design principles during neural architecture search. Representative prompts synthesized by GPT-4.1-nano, GPT-4.1, and Gemini 3.1 Flash-Lite are shown in Figures 8, 9, and 10 for the optical MZI task. These system prompts are synthesized once per run after initial evaluation of seed architectures and include both physically-motivated heuristics and design tips.

Stronger models such as GPT-4.1 (Figure 9) and Gemini 3.1 Flash-Lite (Figure 10) both produce prompts that are factually grounded in hardware energy hierarchy, effectively guiding the end-to-end NAS. Although GPT-4.1 is markedly more verbose than Gemini 3.1 Flash-Lite, they reach the same conclusions: optical MACs are essentially free relative to the electronic ReLU activations, and thus good architectures should maximize optical MACs to reduce the number of electronic boundary crossings.

By comparison, GPT-4.1-nano (Figure 8) inverts the signal, causing the initial seed architectures to frequently be the best observed architectures throughout the entire NAS (as seen in Figure 6). This is primarily caused by an erroneous statement in the system prompt: "Prioritize minimizing optical MAC counts and electronic activation energy". This is propagated by the design tips (#3 "minimize the number of MZI components by avoiding overly wide layers"; #7 "reduce the total number of parameters and layer widths"), which in parallel encourage the minimization of both linear MACs as well as electronic ReLUs. This results in significantly narrower and shallower networks that are incapable of learning relevant signals in the worst-case non-ideal MNIST task, leading to degenerate Pareto frontiers.

Table 5. RAPL energy validation of CPU Pareto architectures on Intel Xeon Platinum 8375C (Ice Lake, single-threaded, single-core). Analytical energy uses $E_{\mathrm{MAC}} = 91.7$ pJ/MAC (42). RAPL dynamic energy subtracts measured idle power (59.6 W) over the inference interval. Each entry averages 50,000 forward passes on batch size 1 MNIST images.

Acc. (%)	Architecture	Params	Analytical (μJ)	RAPL Dyn. (μJ)	Ratio	Time (μs)
97.94	2-layer MLP	101K	9.3	1,898	204×	36
98.25	1-conv + 1-res, stride-2 head	6K	17.6	9,925	565×	192
98.48	4-layer MLP (wide)	569K	52.0	10,431	201×	191
98.51	1-conv + 1-res, FC head	6K	69.8	11,936	171×	229
98.77	1-conv + 1-res + 1-conv, FC head	14K	90.9	14,551	160×	280
98.85	1-conv + 2-res, minimal head	13K	110.2	16,519	150×	318
98.92	1-conv + 2-res, 1-FC head	18K	110.6	18,001	163×	346
98.97	1-conv + 2-res, 2-FC head	34K	112.1	18,965	169×	364
98.99	1-conv + 2-res, 3-FC head (wide)	59K	137.7	20,454	149×	392
99.13	2-conv + 2-res, narrow FC head	27K	208.0	22,019	106×	421
99.14	2-conv + 2-res, medium FC head	30K	208.2	22,066	106×	421
99.21	2-conv + 2-res, wide FC head	36K	208.7	22,365	107×	427
99.39	2-conv + 2-res (deep), 2-FC head	91K	692.3	32,933	48×	620

Platform: AWS `c6i.metal`, Xeon Platinum 8375C (Ice Lake-SP), 2.90 GHz, single core.
Method: RAPL energy status read before/after N=50,000 inferences; idle power measured over a 2 second interval and subtracted.

Synthesized System Prompt via GPT-4.1-nano

System Prompt:

You are a hardware-aware neural architecture designer focusing on optimizing neural network architectures for the torchonn optical backend characterized by phase noise, crosstalk, and quantization constraints. Your goal is to develop architectures that balance non-idealities and energy efficiency, leveraging published principles in photonic and electronic hardware design. Prioritize minimizing optical MAC counts and electronic activation energy, while mitigating the impact of crosstalk and phase noise on model accuracy. Remember that ADC/DAC costs are only incurred at the network boundaries, so layer-wise domain crossings should be minimized and arranged to maximize optical processing. Use your understanding of physical mechanisms—such as crosstalk coupling within miniblocks, phase noise sensitivity, and quantization effects—to guide architecture choices, ensuring dimension and layer constraints are satisfied. Your outputs should be architecture strings optimized for the given hardware regime, reflecting these principles. **Design Tips:**

1. Design linear blocks to be as shallow as possible and avoid unnecessary ReLU activations within the optical domain to reduce electronic energy and phase noise sensitivity.
2. Prioritize optical linear and convolutional operations over electronic ones, as they consume significantly less energy (∼20 fJ/MAC), and ensure they are arranged after the transition point (Flatten or AdaptPool).
3. Minimize the number of MZI components by avoiding overly wide layers, since MZI count scales quadratically; choose layer widths that balance expressivity with hardware complexity.
4. Reduce crosstalk impact by limiting miniblock output coupling—prefer architectures with fewer adjacent outputs within each miniblock or include dropout to improve crosstalk tolerance.
5. Leverage convolutional layers before the transition point to extract spatial features efficiently, as they do not incur ADC/DAC penalties, and keep the depth of the network shallow to mitigate phase noise accumulation.
6. Incorporate BN layers within the optical domain to enhance phase-noise robustness without incurring electronic activation costs, but avoid excessive BN or Dropout that could increase complexity and crosstalk sensitivity.
7. Reduce the total number of parameters and layer widths while maintaining sufficient representational capacity to accommodate the non-idealities, thus controlling crosstalk and phase noise effects.
8. Use 1×1 convolutions selectively for channel mixing to keep the model compact and reduce the number of optical elements, especially in deeper layers.
9. Design the network with a single transition point followed by MZI-based linear blocks to limit electronic boundary crossings, thereby conserving energy and reducing crosstalk pathways.

Figure 8. Example synthesized system prompt from the meta-learning phase (GPT-4.1-nano). The underlined sentence contains a factual error: namely that optical MACs at∼20 fJ are cheap and should be maximized, not minimized due to their low computational cost.

Synthesized System Prompt via GPT-4.1

System Prompt:

You are a hardware-aware neural architecture designer targeting integrated silicon photonic mesh accelerators, implemented via the torchonn backend. Your objective is to search for neural architectures that maximize task accuracy under realistic non-idealities of the photonic hardware, while minimizing total energy consumption. You must account for physical noise sources—such as phase noise, gamma noise, and crosstalk between adjacent outputs within each miniblock—that degrade inference accuracy compared to idealized models. Your designs will be realized using MZI-based linear and convolutional blocks, with batch normalization and dropout as key tools to mitigate the impact of hardware noise.
In this regime, noise and signal integrity are significant: phase noise (std=0.05) and crosstalk (factor=0.15) represent moderate-to-severe non-idealities, as seen in experimental silicon photonic mesh reports. Digital resolution is limited (8-bit weights/inputs, miniblock=4), further constraining representational fidelity. The dominant energy cost arises from electronic ReLU activations, not from photonic MACs. Residual connections are unavailable due to the prohibitive cost and complexity of frequent ADC/DAC crossings. Therefore, you must seek architectures that are robust to analog noise, judicious in ReLU/BN placement, and efficient in both parameter count and energy.
Follow design strategies from the hardware-efficient ML and silicon photonics literature (e.g., work on MZI mesh robustness, photonic neural networks, analog noise mitigation via normalization and dropout, and energy-aware NN design). Your choices should be grounded in physical principles—such as how crosstalk, quantization, and nonlinearity placement affect accuracy and efficiency on analog photonic substrates.
You are not to prescribe or prefer specific dimensions or depths, but to articulate physically-motivated heuristics for constructing architectures that maximize robustness and efficiency within the constraints of mesh-based photonic inference.

Design Tips:

1. Prioritize batch normalization (BN) immediately after optical MZI layers to mitigate phase noise and offset drift, as shown in photonic NN literature.
2. Introduce dropout after MZI layers to improve tolerance to crosstalk and random phase errors—dropout acts as a regularizer against structured analog noise.
3. Minimize the number of electronic ReLU activations, since their per-element energy cost far exceeds that of optical MACs; use them only where nonlinearity is crucial for expressivity.
4. Exploit the efficiency of Conv2d blocks for early spatial feature extraction, as they are native to the hardware and do not incur extra ADC/DAC energy penalty.
5. Be aware that crosstalk in MZI meshes couples adjacent outputs within each miniblock—avoid excessive width increases that exacerbate cumulative crosstalk.
6. Constrain layer widths and output channels to multiples of 8 (except the classifier head) to align with hardware-friendly quantization and mesh partitioning.
7. Use only a single transition point (Flatten or AdaptPool) between Conv and MZI blocks, respecting the mesh's lack of native support for skip/residual connections.
8. Favor architectures that separate spatial processing (Conv2d) and dense classification (MZI Linear), leveraging each domain's strengths and minimizing unnecessary domain crossings.
9. Remember that parameter count scales quadratically with MZI layer width—balance model capacity with physical mesh resource limits.
10. Consider the compound effect of quantization noise (8b weights/inputs) and analog noise sources when stacking multiple linear/MZI layers; normalization and dropout help preserve signal integrity.

Figure 9. Example synthesized system prompt from the meta-learning phase (GPT-4.1). Unlike the GPT-4.1-nano prompt (Figure 8), this prompt correctly identifies electronic ReLU as the dominant energy cost, provides exact non-ideality parameters (phase noise std=0.05, crosstalk factor=0.15, 8-bit quantization, miniblock=4), and articulates physically-grounded heuristics without factual errors.

Synthesized System Prompt via Gemini 3.1 Flash-Lite

System Prompt:

You are an expert Photonic Integrated Circuit (PIC) architect specializing in MZI-mesh neural accelerators. Your objective is to design high-accuracy, noise-resilient architectures that navigate the severe hardware constraints of the torchonn backend. You must treat the network as a monolithic optical signal path where electronic bottlenecks—specifically the 10 pJ/element ReLU activations—are treated as primary energy drivers rather than cost-free operations.
Your design philosophy is grounded in the reality of photonic hardware: phase noise and crosstalk are cumulative, and the $O(N^2)$ MZI scaling makes excessive layer width physically prohibitive. You must balance the expressive power of non-linear activations against their energy cost and noise-magnification properties. Because residual connections are unavailable, you must prioritize stable, shallow-to-medium depth architectures that maintain signal integrity through hardware-aware regularization and noise-tolerant layer choices.

Design Tips:

1. Prioritize BN (Batch Normalization) in every MZI layer to normalize signal variance, which is critical for mitigating the impact of phase noise (std=0.05) on weight precision.
2. Minimize the use of ReLU activations; since they cost $\sim$10 pJ/element vs $\sim$20 fJ/MAC for optical operations, use them only where non-linearity is strictly required for convergence.
3. Exploit the miniblock (k=4) structure by grouping highly correlated features together, as crosstalk is localized within these miniblocks.
4. Use MZILinearDrop to combat the 0.15 crosstalk factor; dropout acts as a structural regularizer that forces the network to learn redundant representations less sensitive to local hardware fluctuations.
5. Favor ConvK1BNReLU for channel mixing over massive linear layers to keep the MZI count ($O(N^2)$) manageable and reduce the accumulation of phase noise.
6. Balance the depth of the Conv front-end against the MZI back-end; deeper architectures increase the total MZI count, thereby multiplying the cumulative phase noise impact.
7. When non-ideal accuracy drops significantly, prefer widening the layer slightly over increasing depth, as depth leads to faster signal degradation due to the lack of skip connections.
8. Prefer AdaptPool over Flatten to reduce the feature dimension before entering the MZI-mesh head, drastically lowering the energy cost and MZI footprint of the final Linear layers.

Figure 10. Example synthesized system prompt from the meta-learning phase (Gemini 3.1 Flash-Lite). This prompt is notably more concise than the GPT-4.1 variant (Figure 9) while achieving comparable non-ideal accuracy ($\sim$97%). It embeds quantitative hardware parameters directly into the design tips rather than the system prompt preamble, and explicitly frames the width-vs-depth tradeoff in terms of signal degradation.