ORIGINAL RESEARCH Open access

PhaseNAS: Large Language Model-Guided Neural Architecture Search for UAV Object Detection

Mohamed Saif[1*], Aisha Ahmed[1], Fatima Al-Mansoori[2], Khalid Al-Hassan[3], Sara Al-Kaabi[4]

Abstract

Neural Architecture Search (NAS) is essential for building perception models that are both efficient and robust for UAV- and drone-based applications, where the key challenges stem from real-time small-object detection and limited hardware resources. Present NAS techniques, including recent ones that harness large language models (LLMs), commonly face shortcomings in static resource provisioning and vague architecture outputs, hampering their utility in rapidly changing aerial contexts. This paper introduces PhaseNAS, an adaptive NAS framework steered by LLMs and purpose-built for drone perception workloads. PhaseNAS dynamically allocates LLM capacity between broad exploration and focused refinement phases, while a newly proposed structured template language translates natural-language instructions into runnable model code. Additionally, we propose a zero-shot detection metric to filter candidate YOLO-based architectures without exhaustive training. Empirical results on NAS-Bench-Macro, CIFAR-10/100, COCO, and VisDrone2019 confirm that PhaseNAS reliably uncovers stronger architectures, slashing search time by up to 86% while simultaneously improving accuracy and resource efficiency. On UAV detection benchmarks, PhaseNAS delivers YOLOv8 variants that achieve higher mAP while maintaining a reduced computational footprint, confirming its readiness for real-time on-board operation. These outcomes imply that PhaseNAS constitutes a practical and broadly transferable solution for autonomous AI model engineering in next-generation UAV platforms.

Keywords Deep learning, Object detection, Large language models, UAV, Neural architecture search, Drone perception

*Correspondence:
Mohamed Saif
mohamed.saif@ku.ac.ae
[1] Department of AI for Hydroponic and Vertical Farming Systems, College of Agriculture, Khalifa University, Abu Dhabi, United Arab Emirates
[2] Department of Precision Water and Nutrient Recycling Analytics, College of Agriculture, UAE University, Al Ain, United Arab Emirates
[3] Department of Machine Learning for Crop Selection in Hyper-Arid Zones, College of Agriculture, American University of Sharjah, Sharjah, United Arab Emirates
[4] Department of Smart LED Light Spectrum Optimization, College of Agriculture, University of Dubai, Dubai, United Arab Emirates

Introduction

Object detection is a core enabler for autonomous drones and unmanned aerial vehicles (UAVs), enabling real-time scene understanding across domains such as aerial surveillance, traffic observation, precision farming, and emergency response [1, 2]. In contrast to ground-level detection, airborne perception imposes stringent, quantifiable demands from both the data and platform domains. To begin with, aerial objects are predominantly small and densely packed. On VisDrone2019 (a standard UAV evaluation suite), a considerable share of target instances reside in small-object categories (e.g., shorter sides measuring 8–32 pixels), making mAP acutely sensitive to multi-scale feature fidelity and partial occlusions [3-5]. Notably, recently introduced UAV-oriented YOLO derivatives purposefully embed P2 small-object detection heads, streamlined neck/backbone designs (e.g., Ghost/EMA), and deformable detection heads to lift VisDrone-

This is an open access article distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0), which permits non-commercial use, distribution, reproduction, and adaptation in any medium or format, provided that appropriate credit is given to the original author(s) and source, a link to the license is provided, and any changes made are indicated. Adapted material must be distributed under the same license.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

like small-object accuracy while respecting edge-device limits [6, 7]. Second, the computational and memory resources available on board are heavily capped [8]. In fielded systems, embedded SoCs (e.g., from the Xavier NX/Orin Nano tier) routinely operate under 5–15 W thermal design power, coupled with stringent model-and-buffer memory ceilings (e.g., < 512 MB) and latency mandates targeting 30–50 FPS (corresponding to ≤ 20–33 ms per frame) to preserve stable UAV flight-control feedback [9, 10]. In practice, lightweight UAV detectors achieve model footprints of just a few megabytes and reach 50–160+ FPS on consumer GPUs by incorporating P2 heads, discarding large-object heads, and adopting mobile-oriented backbones [6, 7]. Third, aerial imagery exhibits pronounced illumination swings, viewpoint and altitude shifts, and persistent occlusions, collectively straining detector consistency and inter-frame association. Such occlusion/disappearance events are endemic in UAV video feeds and routinely demand re-identification or recovery mechanisms within downstream tracking modules [11]. These precisely described constraints drive our objective of automating detector synthesis that prioritizes high small-object accuracy under hard latency/efficiency guarantees, rather than purely tuning generic image classifiers.

Deep learning, especially convolutional neural networks (CNNs), has driven significant progress in UAV object detection, leading to a family of domain-specific architectures, including YOLOv8, EdgeYOLO, and SL-YOLO [9, 10, 12]. Nevertheless, manually engineering and calibrating optimal detection topologies for heterogeneous, non-stationary UAV settings remains a laborious, inherently suboptimal undertaking, particularly given the swiftly evolving trilemma of efficiency, accuracy, and real-time responsiveness.

Neural Architecture Search (NAS) mechanizes the identification of high-performing model topologies, charting a viable course toward bespoke, deployment-grade detection networks [13, 14]. Yet classical NAS paradigms—spanning evolutionary strategies [15], policy-gradient reinforcement learning [16], and differentiable architecture optimization [17]—are frequently computationally prohibitive for UAV-class hardware and are seldom tailored to the peculiarities of small-object detection or edge inference.

A recent line of work has begun enlisting LLMs to (i) propose architecture schematics from natural-language specifications, (ii) emit executable code artifacts, and (iii) close the loop via scoring-driven iteration [18-22]. Despite these advances, the majority of studies remain confined to classification search spaces and employ a uniform LLM footprint throughout, a strategy that proves wasteful during early-stage broad exploration and under-expressive during late-stage fine-tuning. To bridge these salient gaps, we propose PhaseNAS, a dynamic LLM-driven NAS framework architected to meet the exacting constraints of UAV-centric object detection. PhaseNAS weaves together phase-cognizant LLM resource orchestration and template-governed architecture synthesis, enabling sample-efficient, robust discovery of high-caliber detection networks under tight resource constraints. Within UAV small-object regimes, PhaseNAS's detection-tuned zero-shot (ZS) proxy score preferentially promotes high-resolution detection heads and lean neck modules within strict latency–size envelopes, thereby enabling rapid pre-training candidate sieving. This predisposition resonates with contemporary UAV detectors that append P2 heads, drop P5 pathways, and deploy lightweight/mobile backbones to handle small targets [6, 7], as well as with UAV-minded NAS initiatives that jointly balance mAP, throughput, and model compactness [23]. Beyond the model design stage, operational UAV deployments must additionally contend with UTM/airspace safety verification protocols, communication link resiliency, and cyber–physical adversarial threats [24]. Mission-level orchestration alongside ground-based assets and low-altitude economic frameworks further modulate perception requirements and real-time scheduling windows [25]. At the swarm scale, inter-agent communication dropouts, mutual occlusions, and synchronization drift further amplify the robustness demands placed on on-board detectors [26].

While the scoring methodology and evaluation protocol underpinning our NAS framework are grounded in classification-centric research, we systematically extend and recalibrate them for the multi-scale object detection challenges characteristic of UAV environments. For completeness and to exhibit generality, we likewise benchmark compact classification models (under 1 M parameters) as stress tests for severely resource-constrained on-device scenarios. Nonetheless, the principal thrust of this investigation remains the progression toward fully automated, high-performance object detection tailored specifically for UAVs. **Table 1** catalogues representative LLM-NAS approaches. Set against them, our PhaseNAS offers phase-adaptive resource tiering (a lightweight LLM for breadth-oriented exploration; a high-

© 2023 The Author(s). | Published by Agro Intelligence Research Press

capacity LLM for granular refinement), a structured template-to-code transduction layer that curtails malformed generations, and a detection-conscious zero-shot scorer to triage YOLO-style candidates under UAV-specific constraints rapidly.

Table 1. Representative LLM-guided NAS approaches and comparison. “Det.” indicates explicit object detection support; “Dyn.” indicates dynamic LLM capacity/resource adaptation; “Template” indicates explicit, parsable architecture templates; “ZS-Score” indicates training-free/zero-shot scoring.

Method	Core Paradigm	Tasks	Det.	Dyn.	Template	ZS-Score
LLMatic [18]	LLM + QD search	Cls.	No	No	No	No
EvoPrompting [19]	Evo-prompting	Cls.	No	No	No	No
GPT-NAS [27]	GPT + EA-guided NAS	Cls.	No	No	No	No
LLM-GE (GECCO’24) [28]	LLM-guided evolution + source-level code edits	Cls.	No	No	No	No
LeMo-NADe [20]	Code-level param NAS	Cls.	No	No	Yes	No
GUNIUS [21]	Agentic LLM NAS	Cls.	No	No	Yes	No
LLM-GE (GECCO’25) [29]	LLM-guided evolution + YOLO YAML	Det.	Yes	No	Yes	No
PhaseNAS (Ours)	Phase-aware LLM NAS	**Cls. + Det.**	Yes	Yes	Yes	Yes

The central contributions of this work are enumerated below.

1. Phase-aware dynamic LLM allocation: We advance a resource-adaptive NAS methodology that modulates LLM capacity contingent on the search stage, harmonizing expansive exploration with meticulous refinement for UAV detection workloads.

2. Structured detection architecture templates: We craft a parameterized templating scheme that faithfully transduces LLM prompts into executable YOLO-style detection topologies, suppressing erroneous and malformed code outputs across the search loop.

3. Zero-shot detection scoring: We devise a training-free detection-evaluation proxy for rapid, reliable assessment of candidate detectors on UAV corpora such as VisDrone2019, markedly compressing search timelines by obviating exhaustive model retraining.

4. Comprehensive UAV evaluation: We substantiate PhaseNAS’s effectiveness across both classical classification benchmarks (with lightweight model regimes) and demanding UAV detection tasks, registering state-of-the-art mAP and resource efficiency on datasets including COCO and VisDrone2019.

By combining LLM-driven reasoning, adaptive resource scheduling, and robust code synthesis, PhaseNAS advances the automation and intelligence of perception model engineering for next-generation UAV and drone ecosystems, enabling real-time small-object detection that is increasingly potent and accessible for authentic field deployment.

Materials and Methods

© 2023 The Author(s). | Published by Agro Intelligence Research Press

The PhaseNAS framework arises from the pressing need for automated neural architecture search specifically tuned for UAV-based object detection, a domain where precise small-target localization, real-time throughput, and stringent platform constraints are indispensable. Our technique extends zero-shot neural architecture scoring procedures first devised for classification problems, recasting and specializing them to address the distinctive multi-scale, resource-sensitive demands of object detection in aerial contexts. While PhaseNAS is architected as a general-purpose NAS framework capable of serving both object detection and classification, its primary focus is on discovering compact, high-accuracy detection models suitable for UAV fielding. Lightweight classification workloads are incorporated as auxiliary testbeds to probe the NAS efficiency and breadth, particularly for ultra-compact models appropriate for edge or on-board execution.

The foundational realization animating PhaseNAS is that neural architecture search manifests phase-contingent computational needs. Initial exploration thrives on swift, coarse-grained traversal of the search landscape—an undertaking that compact language models can discharge economically. Later-stage polishing demands nuanced deliberation over architectural trade-offs and fine-grained refinements—calling for the sophisticated faculties that exclusively large-scale models possess.

From this observation emerges our governing design axiom: dynamically calibrate computational provisioning to the intricacy of each search phase. Rather than relying on a monolithic LLM throughout the entire search cycle, PhaseNAS fluidly alternates between suitably scaled models, guided by live evaluation of search progress.

As shown in **Figure 1**, the algorithm's key novelty is its adaptive model-sizing scheme: the exploration stage uses a lean, low-overhead language model to conduct expansive and efficient architectural exploration, dramatically compressing search costs. Once high-potential architectural candidates materialize, the refinement stage hands off to a more powerful, more proficient language model, which then polishes and variegates these candidates through advanced reasoning and generative creativity. This bifurcated arrangement strikes a productive compromise between search frugality and the caliber and inventiveness of the terminal architectures, orchestrated by live score thresholds that trigger phase switching and halting.

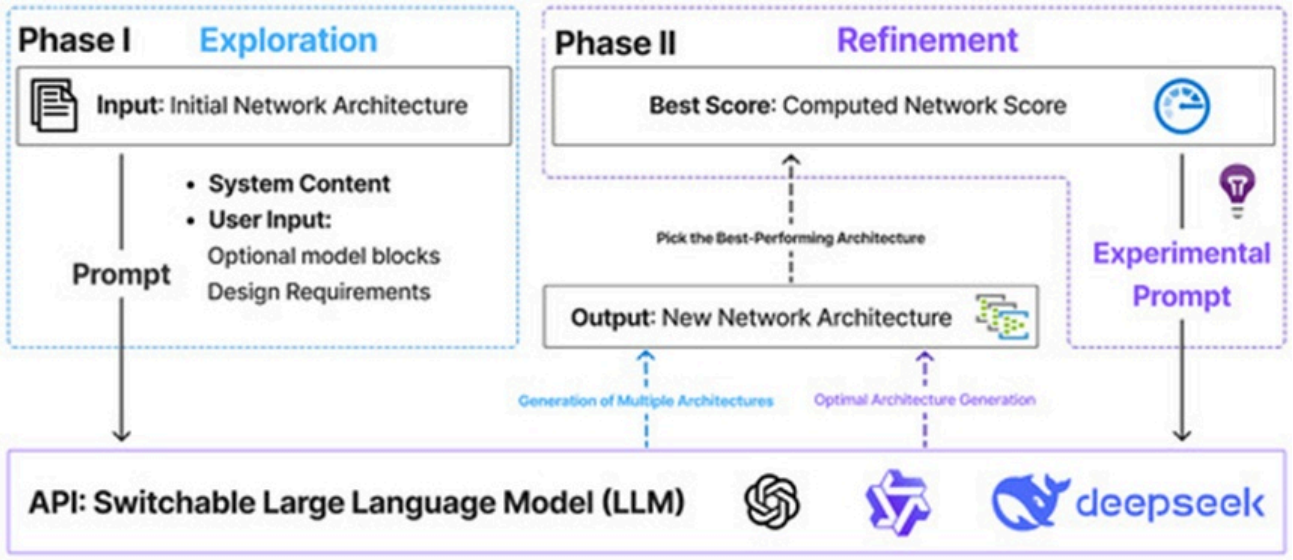

Figure 1. Overview of PhaseNAS. Phase I enlists a small LLM for cost-efficient exploration; Phase II transfers to a larger LLM for high-quality refinement. Phase transitions are triggered by real-time zero-shot scores, harmonizing model capacity with phase complexity to balance efficiency and final quality.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Furthermore, to support more challenging undertakings beyond vanilla classification, PhaseNAS integrates object detection via the YOLOv8 ecosystem, which plays a vital role in delivering real-time scene interpretation and situational awareness aboard autonomous intelligent platforms. This augmentation spotlights the malleability of our approach, bending the search procedure to meet the strict requirements of detection workloads—namely, minimal latency, reduced FLOPs, and tight model footprints.

Dynamic search process

The search procedure unfolds within a restricted space $\mathbb{S}$ delimited by three cardinal design rules:

1. Modular component selection: Every architectural constituent must be drawn from predetermined functional ensembles $\mathbb{G}k$ (e.g., convolutional layers, residual blocks).

2. Dimensional compatibility: Contiguous modules must sustain dimensional coherence through exact channel alignment, ensuring frictionless concatenation.

3. Hardware-aware constraints: The computational burden of the produced architectures must remain within prescribed ceilings, guaranteeing practical viability across heterogeneous deployment contexts.

The methodology advances across two separate stages, each engineered to extract maximal efficiency and performance by harnessing the comparative advantages of dissimilar LLM scales.

Exploration phase

Throughout the exploration stage, a compact, computationally inexpensive language model is tasked with efficiently generating diverse architectural variants. The model ingests natural-language prompts that specify dimensional bounds, complexity caps, and task-specific directives. The resultant candidates are subsequently scored by an evaluation function and screened against resource ceilings. Viable candidates are inserted into a quality-ranked repository, curating the foremost K architectures.

Refinement phase

The moment any candidate housed in the repository clears the transition threshold, the system pivots to the refinement stage. Within this stage, a substantially larger and more proficient language model is brought to bear, further honing and diversifying the high-promise architectures informed by performance diagnostics. The refinement effort aims to upgrade deficient substructures while safeguarding the overall topological skeleton. The search halts upon reaching the stopping threshold, securing methodical traversal of the search manifold.

Search space definition

The conception of an efficacious NAS framework hinges pivotally on a rigorously delineated search space for separate tasks. Within PhaseNAS, the search space is systematically sculpted for both image classification and object detection to enhance efficiency, interoperability, and design novelty simultaneously.

Search space for classification tasks: For classification, the search space $\mathbb{S}$cls is rendered transparent and tightly regimented to streamline search and curtail semantic imprecision. The central stipulations are itemized below:

1. Available building blocks: Each network is pieced together from a prespecified repertoire of residual and convolutional blocks spanning varied kernel dimensions and activation functions. The search is confined to permutations of these standardized primitives, ensuring cross-candidate compatibility and reproducibility.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

2. Channel compatibility: The output channel count of any given block must equal the input channel count of its immediate successor, thereby preserving uninterrupted tensor flow.

3. Input/output constraints: The model ingress must be a trichromatic image, and the egress must be conformant with the classification head.

4. Resource constraints: The aggregate parameter budget and network FLOPs are clamped within deployment-realistic ranges to maintain efficiency.

This transparent, template-grounded formulation ensures that LLMs can faithfully propose, parse, and mutate architectures with negligible risk of ill-formed structures or compilation faults.

Search space for object detection tasks: We ground our work on Ultralytics YOLO, a swift, precise, and accessible model lineage covering detection, tracking, segmentation, classification, and pose inference. For object detection, the search space $\mathbb{S}$det is conceived to broaden and augment the YOLOv8 model family while obeying stringent resource caps. The salient properties are enumerated below:

1. YOLOv8 as baseline: The search is moored on the stock YOLOv8n and YOLOv8s backbones, sanctioning reconfiguration and rearrangement of backbone, neck, and head blocks, while restraining total parameters and FLOPs to remain commensurate with the reference models.

2. Block-level modifications: Candidate architectures arise from reshuffling, swapping, or interposing blocks within the backbone and neck, provided that input/output channel dimensionalities and tensor geometries remain mutually consistent.

3. Innovative block extensions:

· YOLOv8+: Instances under this heading are spawned by reshuffling extant YOLOv8 blocks and fine-tuning their hyperparameters, abstaining from the introduction of unfamiliar block categories.

· YOLOv8: Over and above the liberties accorded to YOLOv8+, this heading expressly ushers in two unprecedented blocks:

- SCDown [30]: A spatial-compression downsampling unit engineered to fortify multi-scale feature extraction and ameliorate information propagation through the early backbone.

- PSA [30]: A polarized self-attention module that injects channel-wise attention to sharpen target localization and feature discrimination.

These purpose-built blocks are exclusively accessible within the YOLOv8 search space, endowing the framework with the license to probe more expressive and potent topologies that transcend the stock YOLOv8 repertoire. These novel modules carry particular salience for UAV settings, wherein the faculty to safeguard fine-grained representations and amplify small-object localization under pinched computational envelopes is paramount.

4. Multi-Scale and Detection Head Compatibility: Every searched architecture must furnish multi-scale feature pyramids and remain interoperable with the YOLO detection heads.

5. Resource Constraints: Parameter tallies and FLOPs are confined within the native YOLOv8n/s envelopes to assure real-time operability and equitable benchmarking.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Summary: The detection search space, while anchored in YOLOv8, is enriched by integrating two sophisticated modules (SCDown and PSA), conferring greater architectural breadth and enabling higher mAP under resource constraints. As summarized in **Table 2**, we articulate a nested search topology: YOLOv8+ enforces block-level reshuffling within the YOLOv8 lineage, whereas YOLOv8 additionally enables SCDown/PSA toggles while preserving identical n/s resource allowances. The classification search space is rigorously bounded to 10 modular blocks, each endowed with explicit parameterizations and depth ceilings.

Table 2. Compact comparison of YOLOv8 variants in the detection search space. Budget: observance of YOLOv8n/s params/FLOPs. (+) denotes in-family reconfiguration; () appends SCDown/PSA while honoring identical n/s budgets.

Model configuration	Coverage	Main modifications	Additional components
YOLOv8 (reference model)	Standard official n/s backbones with conventional neck/head and multi-scale outputs (P3–P5)	None	No
YOLOv8+ (restructured)	Internal block restructuring across the backbone and neck	Rearranged and fine-tuned blocks while preserving input/output dimensions	No
YOLOv8* (enhanced)	YOLOv8+ integrated with optional add-on modules	SCDown and PSA are enabled selectively for each candidate	Yes

Algorithm overview

Algorithm 1 sketches the full PhaseNAS search workflow, adaptively switching between LLM-guided exploration and streamlined refinement based on live score benchmarks. Regarding cost containment, PhaseNAS confines large-LLM engagement by dispatching small-LLM queries during early-stage discovery and activating the large LLM only once the transition criterion is met, thereby packing high-capacity reasoning into the terminal < 50% of iterations observed in our experiments.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Require: Initial architecture S_{init}, small LLM $\mathcal{ME}$ (for Exploration), large LLM $\mathcal{MR}$ (for Refinement), thresholds γ_{trans},γ_{stop}, pool size K
Ensure: Optimal architecture $S*$

Initialize: $\mathbb{A}c \leftarrow \{S_{init}\}$, $\phi \leftarrow$ Exploration Phase
While $\max_{S \in \mathbb{A}c} Ez(S) < \gamma_{stop}$ **do**
  if ϕ = Exploration **then**
    Generate $S_{new} \leftarrow \mathcal{ME}$ (exploration_prompt) ▷ Small LLM for cost-efficient exploration
    if $\mathcal{V}(S_{new})$ and $Ez(S_{new}) > \min_{S \in \mathbb{A}c} Ez(S)$ **then**
      Add S_{new} to $\mathbb{A}c$
      if $|\mathbb{A}c| > K$ **then**
        Remove architecture with the lowest Ez from $\mathbb{A}c$
      end if
    end if
    if $\exists S \in \mathbb{A}c$ such that $Ez(S) \geq \gamma_{trans}$ **then**
      $\phi \leftarrow$ Refinement Phase
      Set base architecture $S_{base} \leftarrow \arg\max_{S \in \mathbb{A}c} Ez(S)$
    end if
  else ▷ Refinement Phase
    Generate refined architecture $S_{new} \leftarrow \mathcal{MR}$ (S_{base},feedback) ▷ Large LLM for high-quality refinement
    if $\mathcal{V}(S_{new})$ and $Ez(S_{new}) > Ez(S_{base})$ **then**
      Update base architecture: $S_{base} \leftarrow S_{new}$
      Add S_{new} to $\mathbb{A}c$
    end if
  end if
end while
return $S* \leftarrow \arg\max_{S \in \mathbb{A}c} Ez(S)$

Algorithm 1. PhaseNAS Architecture Search

In our setup, we specify $\mathcal{ME}$ with several choices, spanning Qwen2.5-7B, Qwen2.5-14B, and Qwen2.5-32B (7–32 billion parameters) for thrifty exploration, and $\mathcal{MR}$ as Qwen2.5-72B, Llama-3.3-70, and Claude-3.5-Sonnet (70+ billion

© 2023 The Author(s). | Published by Agro Intelligence Research Press

parameters) for premium refinement. The precise model scales can be tuned to fit the computational budget at hand, with the cardinal rule being $|\mathcal{M}E| < |\mathcal{M}R|$ to enforce progressive capability ramping.

We set phase-change thresholds appropriate to the task. For classification, γtrans is pegged to roughly 80% of γstop (namely, γtrans ≈ 0.8 × γstop), with γstop representing a relatively stringent, fixed-target NAS-score ceiling. For detection, γtrans follows a progress-based rule: the switch fires when the candidate score stagnates across 3–5 consecutive rounds (patience = 3–5), whereas γstop persists as a high, pre-established target. Under sensitivity probing (±5 percentage points around the 0.8 × γstop prescription for classification; patience $\in$ {3,5} for detection; and elevated γstop equivalents spanning the P93–P98 band), the top-5 hierarchy stays robust (Spearman ϱ > 0.9); these knobs primarily control exploration depth and shift timing instead of reshuffling the best candidate.

LLM-compatible architecture representation

A fundamental bottleneck in LLM-based NAS is reconciling natural-language specifications with executable model code [22, 31]. PhaseNAS leverages a parameterized, block-level templating idiom that explicitly captures kernel sizes, channel depths, strides, and skip-connection layouts. As an illustration, a convolutional block reads as ConvK3BNRELU(3, 8, 1, 1) and a residual block as ResK3K3(16, 32, 2, 1). This unambiguous interface drastically reduces vagueness and parsing failures, improves compilation success rates, and slots neatly into modular detection frameworks like YOLOv8. We painstakingly craft the LLM prompts to shepherd the architecture search loop.

Task adaptation: From classification to object detection

To steer the architecture search loop productively, PhaseNAS deploys task-specific NAS scoring routines to evaluate candidate architectures. These metrics are engineered to capture how architectures react to input distortion and how stable their normalization layers remain, delivering quick generalization proxies that sidestep full training cycles. Below, we detail the scoring recipes for classification and detection, together with the adjustments introduced for the YOLO ecosystem.

NAS score for classification

We retain the scoring formulation advanced in Zen-NAS [32], which effectively captures model sensitivity to perturbations alongside Batch Normalization layer stability. Since the underlying mathematics stays identical to Zen-NAS, the detailed derivations are omitted here for compactness.

NAS score for object detection

Detection pipelines demand multi-scale feature integrity and elevated input resolutions. We accordingly generalize Zen-style zero-shot scoring to assess the coherence of multi-scale representations under input perturbation, collecting signals across pyramid stages and merging them with BN stability indicators. The result is a detector-tuned surrogate that tracks downstream mAP while obviating full training.

• Input perturbation:

$$x_{\text{mix}} = x_1 + \gamma \cdot x_2. \quad (1)$$

• Feature map extraction:

Let model (·) be given, and let (·) extract internal feature tensors. We determine:

$$f_1 = F(M(x_1)), \quad f_{\text{mix}} = F(M(x_{\text{mix}})), \quad (2)$$

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Where (l) $\in \mathbb{R}$^{$b \times cl \times hl \times wl$} designates the feature map at scale l.

• Multi-scale difference:

The total feature perturbation signal is:

$$\Delta = \sum_{l=1}^{L} \left\| f_1^{(l)} - f_{\text{mix}}^{(l)} \right\|_1, \quad (3)$$

With L representing the quantity of feature scales.

• BatchNorm scaling:

$$\mathcal{B} = \sum_{m} \log \left(\sqrt{\text{running_var}_m + \epsilon} \right). \quad (4)$$

• Final NAS score:

The detection score takes the form:

$$NAS_{det} = \log (\Delta + \epsilon) + \mathcal{B} \quad (5)$$

Rationale: Δ captures the vigor of the model's pyramid feature response (P3–P5) to a gentle input nudge; a larger Δ points to superior feature separation but may also flag instability. $\mathcal{B}$ aggregates BN running variances across the network; larger $\mathcal{B}$ denotes well-distributed activation ranges and settled normalization dynamics. The fused score log((Δ + ϵ) + $\mathcal{B}$) thus endorses architectures that yield discriminating, multi-scale representations while preserving normalization equilibrium. Across the YOLOv8{n, s, m, l, x} family, $NASdet$ exhibits tight alignment with trained mAP (high Spearman ϱ and Pearson r).

• Repetition and aggregation:

Redo the entire process R times and derive:

$$\mu = \frac{1}{R} \sum_{t=1}^{R} s_t, \quad \sigma = \sqrt{\frac{1}{R} \sum_{t=1}^{R} (s_t - \mu)^2}, \quad (6)$$

Where μ serves as the figure of merit for ranking and selection.

Summary: The NAS scoring apparatus for classification and detection provides a cohesive yet flexible framework for cost-effectively evaluating candidate architectures. By stretching the scoring paradigm to encompass multi-scale outputs and detector-centric complexities, PhaseNAS secures durable, scalable neural architecture search across an expansive spectrum of AI workloads.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Results and Discussion

Experimental Design Rationale. Our evaluation blueprint is crafted to verify two claims: that PhaseNAS lowers search cost through dynamic LLM scaling and structured encoding, and that the architectures it uncovers beat leading NAS baselines on both efficiency and accuracy. The primary lens is UAV-based object detection—small-object, real-time regimes captured by VisDrone2019 and COCO. To probe generality and portability, we also include classification tasks on sub-1M-parameter models as auxiliary stress tests for severely resource-constrained deployments.

We first validate PhaseNAS by benchmarking it head-to-head against the recent LLM-driven NAS system GUNIUS on NAS-Bench-Macro. This trial isolates the gain from dynamic, phase-aware scheduling and confirms the value of adaptive model handoff. We then broaden the scope to cover both image classification and object detection, applying PhaseNAS across multiple search spaces and tasks to demonstrate scalability and cross-domain applicability.

Comparison with LLM-based NAS baselines

To isolate the core merit of PhaseNAS, we first pit it against the recent LLM-based NAS method GUNIUS on NAS-Bench-Macro. GUNIUS runs Qwen2.5-32B and Qwen2.5-72B as independent generators, whereas PhaseNAS dynamically switches between them during the search phase.

Experimental SETUP

All contenders are assessed on NAS-Bench-Macro under identical search rules. The published GUNIUS recipe uses GPT-4 as the LLM backbone; for accessibility and reproducibility, we substitute Qwen2.5-32B and Qwen2.5-72B for both GUNIUS and PhaseNAS. Under GUNIUS, each model generates architectures in isolation. For PhaseNAS, the phase controller adaptively oscillates between Qwen2.5-32B and Qwen2.5-72B guided by the candidate score distribution. Every search run spans 10 iterations to ensure convergence.

Results and analysis

As shown in **Figure 2**, all three variants are evaluated under a shared protocol. GUNIUS with Qwen2.5-32B peaks at 92.73% accuracy with a final rank of 119. Switching to Qwen2.5-72B yields a marginal uptick: 92.75% accuracy, rank 110. By contrast, PhaseNAS reaches 93.11% accuracy and a markedly better rank of 3—indicating not just higher accuracy, but solutions positioned much closer to the global optimum in the search space.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

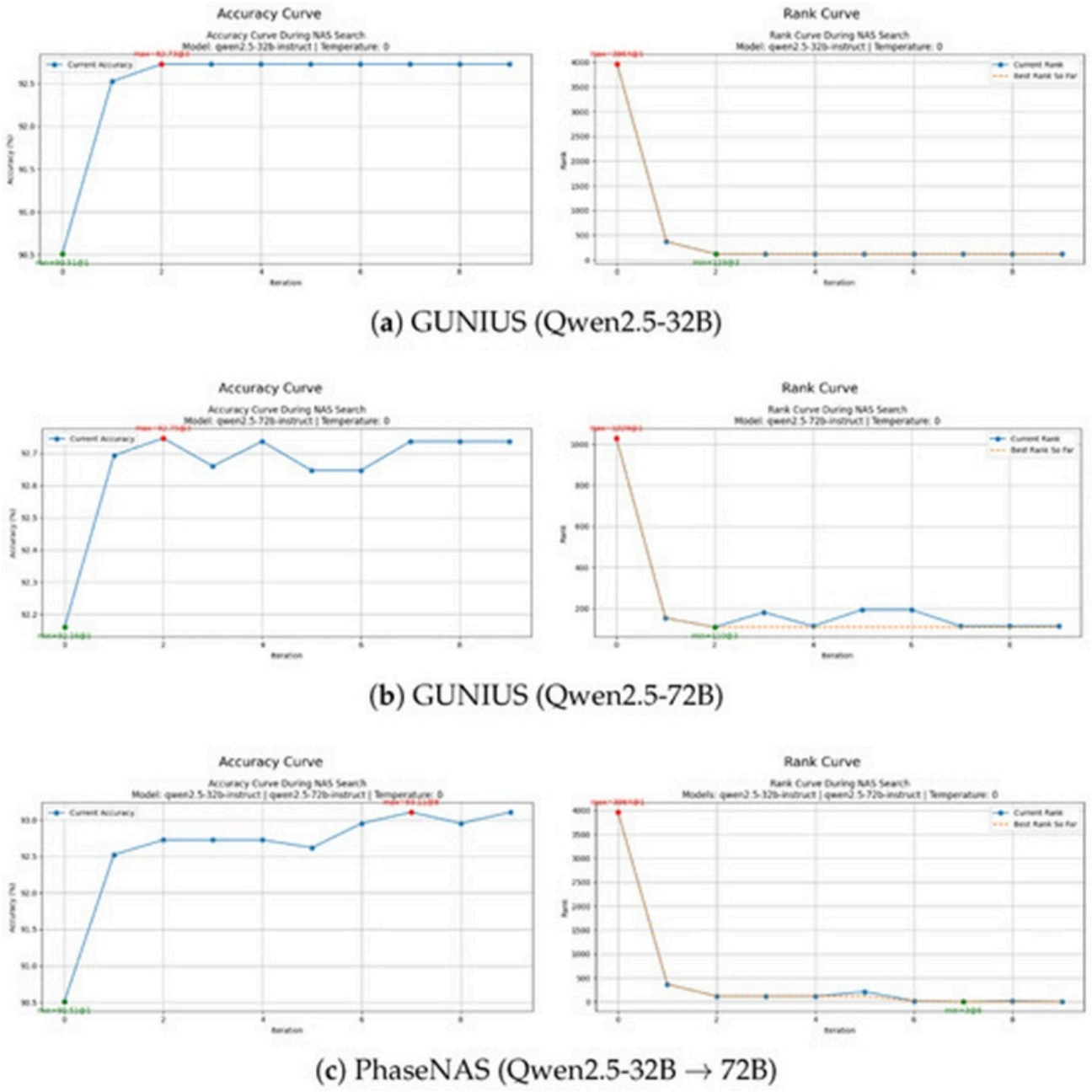

Figure 2. Accuracy and rank trajectories on NAS-Bench-Macro for three approaches.

Cost–benefit note

Relative to single-LLM baselines, PhaseNAS reserves large-LLM queries for late refinement while a small LLM handles breadth-first exploration. In practice, this shrinks exploration-phase LLM consumption and end-to-end wall-clock time while boosting best-found accuracy and rank **(Figure 2)**. This targeted allocation explains the efficiency gains later replicated in classification and detection trials.

Generalization to classification and detection tasks

Building on the LLM-NAS macro-benchmark, we next stress-test PhaseNAS across image classification and object detection to gauge its generalizability and scalability. Classification experiments use CIFAR-10 and CIFAR-100; detection

© 2023 The Author(s). | Published by Agro Intelligence Research Press

experiments run on the COCO validation split. To keep architecture comparisons fair and interpretable, we enforce consistent constraints throughout search:

• Model size: Bound total parameters for edge-device suitability.

• FLOPs: Cap computation to preserve real-time throughput.

• Latency: Safeguard practical inference speeds on reference hardware.

• Depth: Avoid impractically deep, optimization-resistant networks.

All methods use the same NAS scoring functions and aligned training pipelines to ensure fairness.

Hardware/Software: All searches executed on 1× NVIDIA GeForce RTX 4090 24GB, AMD EPYC 7642 CPU, 512 GB RAM, Ubuntu 22.04, CUDA 12.1, Python 3.10.13, PyTorch 2.2, Ultralytics 8.3.63. YOLO training adopted imgsz = 640, batch = 32, AdamW with cosine schedule, 600 epochs (COCO)/200 epochs (VisDrone), default Ultralytics augmentation unless noted. Zero-shot scoring ran on GPU under mixed precision. Reported search times are end-to-end wall-clock inclusive of compilation/validation.

Classification results and analysis

We compare PhaseNAS against the widely used Zen-NAS on CIFAR-10 and CIFAR-100, maintaining fairness by adopting identical Zen-score evaluation. **Table 3** summarizes results across search configurations.

Table 3. PhaseNAS versus Zen-NAS on classification benchmarks. For PhaseNAS, parenthetical values after Search Time denote relative reduction over Zen-NAS under the matching setting. Accuracies reported as mean ± SD across 3 runs.

Method	CIFAR-100 Acc.	CIFAR-10 Acc.	Search Time (min)	Zen-Score
Zen-NAS	79.95 ± 0.08	96.72 ± 0.07	7.43	99.20
PhaseNAS	**80.76 ± 0.07**	**96.80 ± 0.06**	**1.20 (−83.9%)**	**99.43**
Zen-NAS	80.78 ± 0.07	96.00 ± 0.08	19.07	111.63
PhaseNAS	**81.22 ± 0.06**	**96.65 ± 0.05**	**7.06 (−63.0%)**	**111.33**
Zen-NAS	81.24 ± 0.09	96.94 ± 0.06	67.46	121.38
PhaseNAS	**81.35 ± 0.07**	**97.33 ± 0.04**	**9.01 (−86.6%)**	**121.44**

As **Table 3** indicates, PhaseNAS slashes search time by up to 86% compared to Zen-NAS, delivers classification accuracy on par or better (particularly on CIFAR-100), and achieves comparable Zen-scores, pointing to stronger architectural potential.

These outcomes confirm that PhaseNAS can dramatically compress the NAS timeline while holding or lifting final accuracy. The efficiency dividend stems from the dynamic interplay between small and large language models under phase-aware control, as previously validated on the LLM-based NAS macro-benchmark.

Object detection results and analysis

To further stress the versatility of PhaseNAS, we deploy it for object detection on COCO and VisDrone2019 datasets within the YOLOv8 ecosystem. Architecture search respects the same FLOPs and parameter envelopes as YOLOv8n

© 2023 The Author(s). | Published by Agro Intelligence Research Press

and YOLOv8s, with LLM-driven prompts steering both exploration and refinement.

Table 4 reveals that PhaseNAS-produced detectors not only surpass the manually designed YOLOv8 baselines in mAP, but also trim model bulk and computational load on both generic and drone-centric datasets.

Table 4. Architecture comparison on COCO and VisDrone2019. mAP@50:95 given as mean ± sd over 3 seeds.

COCO					
Family	**Model**	**GFLOPs**	**ΔmAP**	**MParams**	**mAP@50:95 (mean ± sd)**
YOLOv8n	YOLOv8n	8.7	–	3.2	37.30 ± 0.05
	YOLOv8n+	8.2 (−5.7%)	**+0.72 (+1.9%)**	3.0 (−6.3%)	**38.02 ± 0.04**
	YOLOv8n*	8.5 (−2.3%)	**+1.78 (+4.8%)**	2.95 (−7.8%)	**39.08 ± 0.07**
YOLOv8s	YOLOv8s	28.6	–	11.2	44.90 ± 0.08
	YOLOv8s+	25.0 (−12.6%)	**+0.52 (+1.1%)**	10.3 (−8.0%)	**45.42 ± 0.05**
	YOLOv8s*	22.4 (−21.7%)	**+1.18 (+2.6%)**	9.9 (−11.6%)	**46.08 ± 0.09**
VisDrone2019					
Family	**Model**	**GFLOPs**	**ΔmAP**	**MParams**	**mAP@50:95 (mean ± sd)**
YOLOv8n	YOLOv8n	8.7	–	3.2	18.50 ± 0.04
	YOLOv8n+	8.2 (−5.7%)	**+0.32 (+1.7%)**	3.0 (−6.3%)	**18.82 ± 0.05**
	YOLOv8n*	8.5 (−2.3%)	**+0.60 (+3.2%)**	2.95 (−7.8%)	**19.10 ± 0.08**
YOLOv8s	YOLOv8s	28.6	–	11.2	22.50 ± 0.07
	YOLOv8s+	25.0 (−12.6%)	**+0.18 (+0.8%)**	10.3 (−8.0%)	**22.68 ± 0.09**
	YOLOv8s*	22.4 (−21.7%)	**+0.88 (+3.9%)**	9.9 (−11.6%)	**23.38 ± 0.10**

ΔmAP denotes absolute mean gain over the family base (YOLOv8n or YOLOv8s); parenthetical percentages indicate relative shift versus the family base for mAP, parameters, and FLOPs (negative = reduction). (+) signifies in-family reconfiguration; () unlocks SCDown/PSA under identical n/s budgets.

COCO results

• YOLOv8n family: YOLOv8n+ and YOLOv8n raise mAP to 38.0 and 39.1, respectively, each with leaner parameters and fewer FLOPs than the stock model (37.3, 3.2 M, 8.7 G).

• YOLOv8s family: YOLOv8s+ and YOLOv8s hit mAPs of 45.4 and 46.1, respectively, while cutting parameter count and FLOPs below the baseline (44.9, 11.2 M, 28.6 G).

VisDrone2019 results

• YOLOv8n family: On the challenging VisDrone2019 benchmark, PhaseNAS pushes mAP@50:95 from 18.5 (YOLOv8n) to 18.8 (YOLOv8n+) and 19.1 (YOLOv8n), staying within tighter parameter and FLOPs budgets.

• YOLOv8s family: Likewise, mAP@50:95 rises from 22.5 (YOLOv8s) to 22.7 (YOLOv8s+) and 23.4 (YOLOv8s), further cementing the value of PhaseNAS in drone-based detection regimes.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Efficiency perspective and UAV relevance. Under YOLOv8n/s budgets, PhaseNAS variants deliver higher mAP with fewer parameters and lower FLOPs, improving accuracy-per-GFLOP and per-MParam figures. This directly translates into stronger real-time candidacy on resource-constrained onboard compute, typical of UAV platforms, where latency and memory are at a premium. The consistent uplift on VisDrone2019—a taxing drone benchmark dense with small objects—underscores that PhaseNAS is well-matched to real-world aerial perception, enabling more accurate and efficient onboard deployment.

The experimental evidence establishes that PhaseNAS achieves superior performance and efficiency compared to both classic and recent LLM-based NAS baselines, particularly for UAV-focused object detection. This advantage stems from three interlocking factors: (1) Adaptive resource orchestration: By fluidly transitioning between small and large LLMs across search phases, PhaseNAS enables rapid, low-cost exploration followed by high-fidelity refinement, substantially compressing total search cost while preserving or elevating final model quality. (2) Structured encoding: Template-grounded architecture specification reduces semantic drift and decode failures, ensuring LLMs emit valid, compilable architectures even for complex detection pipelines. This pain point routinely hampers earlier LLM-NAS schemes relying on free-form prompting. (3) Task-specific scoring: The detection-oriented NAS scoring function capably captures multi-scale feature quality and normalization stability, supplying a rugged, training-free proxy for downstream detection accuracy—a property of heightened importance for small-object, real-time UAV scenarios.

Measured against prior works such as GUNIUS and Zen-NAS, which either lock in a fixed LLM or focus on classification, PhaseNAS offers a phase-cognizant, adaptive resource policy and extends NAS scoring into the detection domain, thereby closing the gap between generic NAS tooling and the strict requirements of UAV perception. PhaseNAS improves the end-to-end cost–benefit equation by (i) trimming search-time LLM and compute expenditure via phase-aware allocation, and (ii) yielding detectors with higher accuracy per FLOP and per parameter suited for real-time UAV deployment. Our VisDrone2019 and COCO results further corroborate that PhaseNAS not only transfers across task types but reliably produces better mAP and lower computational cost—central criteria for real-world drone fielding. These findings have broader significance for the design of automated AI systems in resource-limited, latency-sensitive environments. The dynamic LLM allocation and structured representation mechanisms may apply to other edge AI domains beyond UAVs, including robotics, IoT, and intelligent surveillance.

Our current design deliberately anchors on YOLO-style pipelines because they naturally align with LLM-directed edits through transparent, machine-parseable architecture blueprints and provide a battle-tested deployment ecosystem for real-time UAV operation. This pragmatic commitment narrows the architectural spectrum and complicates out-of-the-box validation on detectors that lack standardized, script-readable descriptors (e.g., SSD, Faster R-CNN). Template-driven, parseable specifications are vital to ensuring candidate outputs remain well-formed and compilable under rudimentary input/output compatibility rules and resource ceilings; nevertheless, they also erect a practical “template boundary” that may skew exploration toward familiar building blocks. We actively cultivate diversity inside this boundary (stochastic decoding, near-duplicate culling, macro-topology mutations). We will broaden reach by (i) formulating compact, parseable descriptors for additional detector lineages and (ii) incorporating an agent-style orchestration layer that manages proposals extending beyond the incumbent template. Finally, we will generalize the template machinery with reusable sanity checks and exemplar-based guidance to smooth transfer across neighboring domains.

In sum, PhaseNAS demonstrates that LLM-powered, resource-adaptive NAS frameworks can effectively automate the engineering of high-caliber, efficient perception models for UAVs and related application areas. We expect that future leaps in language model reasoning and program synthesis will further magnify the autonomy and breadth of neural architecture search, clearing a path toward next-generation intelligent aerial platforms. To recapitulate, PhaseNAS sharpens end-to-end cost–benefit by (i) compressing search-phase LLM and compute outlay through phase-aware scheduling and (ii) yielding detectors that pack greater accuracy per FLOP and per parameter for real-time UAV fielding.

Conclusion

© 2023 The Author(s). | Published by Agro Intelligence Research Press

We introduce PhaseNAS, a neural architecture search framework that addresses the central resource-allocation tension in LLM-based NAS via phase-aware dynamic scaling. By calibrating LLM heft to the requirements of each search stage, PhaseNAS delegates broad, breadth-first exploration to lightweight models and reserves heavy-lifting reasoning for terminal refinement.

Extensive experiments corroborate PhaseNAS across classification and detection regimes: it surfaces stronger architectures on NAS-Bench-Macro (93.11% accuracy, rank 3), reduces wall-clock search time by up to 86% on classification benchmarks, and extends to object detection with automatically composed YOLOv8 variants that eclipse baselines. Under equal parameter and FLOP envelopes, these variants achieve higher mAP while consuming fewer resources, improving accuracy per GFLOP and per million parameters, and reinforcing real-time operability on the resource-tight on-board compute platforms characteristic of UAV systems.

Collectively, PhaseNAS bears out that resource-aware, adaptive search can jointly elevate effectiveness and end-to-end cost–benefit—paring search-time LLM/compute usage while furnishing models with superior accuracy–efficiency profiles at inference. Future efforts will broaden phase-aware allocation to further modalities and tasks, and fold in richer hardware-aware constraints alongside explicit latency targets.

Abbreviations

NAS	Neural Architecture Search
LLM	Large Language Model
FLOPs	Floating Point Operations
mAP	Mean Average Precision
UAV	Unmanned Aerial Vehicle
CNN	Convolutional Neural Network

Acknowledgements

The authors thank all members of Qiyuan Lab for helpful discussions and support.

Conflict of interest

None

Financial support

Qiyuan Lab, grant number S20240201001, funded this research.

Ethics statement

None

Received: 02 May 2022 Revised: 07 Jul 2022 Accepted: 22 Jul 2022
Published online: 25 November 2023

© 2023 The Author(s). | Published by Agro Intelligence Research Press

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at https://creativecommons.org/licenses/by-nc-sa/4.0/.

References

1. Ramachandran A, Sangaiah AK. A review on object detection in unmanned aerial vehicle surveillance. Int J Cogn Comput Eng. 2021;2:84-97.

2. Du D, Zhu P, Wen L, Bian X, Lin H, Hu Q, et al. VisDrone-DET2019: the vision meets drone object detection in image challenge results. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW); Seoul, Republic of Korea; 2019.

3. Wang P, Zhao J. SOD-YOLO: Enhancing YOLO-based detection of small objects in UAV imagery. arXiv. 2025;arXiv:2507.12727.

4. Zheng Y, Jing Y, Zhao J, Cui G. LAM-YOLO: Drones-based small object detection on lighting occlusion attention mechanism YOLO. Comput Vis Image Underst. 2025;235:104489.

5. Lu L, He D, Liu C, Deng Z. MASF-YOLO: An improved YOLOv11 network for small object detection on drone view. arXiv. 2025;arXiv:2504.18136.

6. Huang M, Mi W, Wang Y. EDGS-YOLOv8: An improved YOLOv8 lightweight UAV detection model. Drones. 2024;8:337.

7. Nguyen PT, Nguyen GL, Bui DD. LW-UAV–YOLOv10: a lightweight model for small UAV detection on infrared data based on YOLOv10. Geomatica. 2025;77:100049.

8. Gong J, Liu H, Zhao L, Maeda T, Cao J. EEG-powered UAV control via attention mechanisms. Appl Sci. 2025;15:10714.

9. Liu S, Zha J, Sun J, Li Z, Wang G. EdgeYOLO: an edge-real-time object detector. In: Proceedings of the 42nd Chinese Control Conference (CCC); Tianjin, China. 2023. p. 9130-5.

10. Chen D, Zhang L. SL-YOLO: A stronger and lighter drone target detection model. arXiv. 2024;arXiv:2411.11477.

11. Micheal AA, Micheal A, Gopinathan A, Barathi BUA. Deep learning-based multi-class object tracking with occlusion handling mechanism in UAV videos. Res Sq. 2024. https://doi.org/10.21203/rs.3.rs-4488926/v1.

12. Jocher G, Chaurasia A, Qiu J. Ultralytics YOLOv8 (Version 8.0.0). GitHub Repository. 2023. https://github.com/ultralytics/ultralytics (accessed on 1 November 2025).

13. Elsken T, Metzen JH, Hutter F. Neural architecture search: a survey. J Mach Learn Res. 2019;20:1-21.

14. Real E, Aggarwal A, Huang Y, Le QV. Regularized evolution for image classifier architecture search. Proc AAAI Conf Artif Intell. 2019;33:4780-9.

15. Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, et al. Large-scale evolution of image classifiers. In: Proceedings of the 34th International Conference on Machine Learning (ICML 2017); Sydney, NSW, Australia. https://proceedings.mlr.press/v70/real17a.html (accessed on 1 November 2025).

16. Zoph B, Le Q. Neural architecture search with reinforcement learning. arXiv. 2016;arXiv:1611.01578.

© 2023 The Author(s). | Published by Agro Intelligence Research Press

17. Liu H, Simonyan K, Yang Y. DARTS: differentiable architecture search. arXiv. 2018;arXiv:1806.09055.

18. Nasir MU, Earle S, Togelius J, James S, Cleghorn C. LLMatic: Neural architecture search via large language models and quality diversity optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference; Melbourne, Australia. 2024.

19. Chen A, Dohan D, So DR. EvoPrompting: language models for code-level neural architecture search. arXiv. 2023;arXiv:2302.14838.

20. Rahman MH, Chakraborty P. Lemo-nade: Multi-parameter neural architecture discovery with LLMs. arXiv. 2024;arXiv:2402.18443.

21. Zheng M, Su X, You S, Wang F, Qian C, Xu C, et al. Can GPT-4 perform neural architecture search? arXiv. 2023;arXiv:2304.10970.

22. Wu X, Wu SH, Wu J, Feng L, Tan KC. Evolutionary computation in the era of large language model: survey and roadmap. arXiv. 2024;arXiv:2401.10034.

23. Slimani H, El Mhamdi J, Jilbab A. Deep learning structure for real-time crop monitoring based on neural architecture search and UAV. Braz Arch Biol Technol. 2024;67:e24231141.

24. Asghari O, Ivaki N, Madeira H. UAV operations safety assessment: A systematic literature review. ACM Comput Surv. 2025;57:1-37.

25. Wang Y, Li J, Yang X, Peng Q. UAV–ground vehicle collaborative delivery in emergency response: a review of key technologies and future trends. Appl Sci. 2025;15:9803.

26. Alqudsi Y, Makaraci M. UAV swarms: research, challenges, and future directions. J Eng Appl Sci. 2025;72:12.

27. Yu C, Liu X, Wang Y, Liu Y, Feng W, Xiong D, et al. GPT-NAS: evolutionary neural architecture search with the generative pre-trained model. J Latex Cl Files. 2025;14:1-11.

28. Morris C, Jurado M, Zutty J. LLM guided evolution—The automation of models advancing models. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24); Melbourne, Australia; 2024. p. 377-84.

29. Yu Y, Zutty J. LLM-guided evolution: an autonomous model optimization for object detection. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '25 Companion); Málaga, Spain. 2025. p. 2363-70.

30. Wang A, Chen H, Liu L, Chen K, Lin Z, Han J, Ding G. YOLOv10: real-time end-to-end object detection. arXiv. 2024;arXiv:2405.14458.

31. Dong X, Yang Y. NAS-Bench-201: extending the scope of reproducible neural architecture search. arXiv. 2020;arXiv:2001.00326.

32. Lin M, Wang P, Sun Z, Chen H, Sun X, Qian Q, Li H, Jin R. Zen-NAS: a zero-shot NAS for high-performance image recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; Montreal, BC, Canada. 2021. p. 347-56.

© 2023 The Author(s). | Published by Agro Intelligence Research Press