# COGNITIVE-YOLO: LLM-DRIVEN ARCHITECTURE SYNTHESIS FROM FIRST PRINCIPLES OF DATA FOR OBJECT DETECTION

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

## ABSTRACT

Designing high-performance object detection architectures is a complex task, where traditional manual design is time-consuming and labor-intensive, and Neural Architecture Search (NAS) is computationally prohibitive. While recent approaches using Large Language Models (LLMs) show promise, they often function as iterative optimizers within a search loop, rather than generating architectures directly from a holistic understanding of the data. To address this gap, we propose Cognitive-YOLO, a novel framework for LLM-driven architecture synthesis that generates network configurations directly from the intrinsic characteristics of the dataset. Our method consists of three stages: first, an analysis module extracts key meta-features (e.g., object scale distribution and scene density) from the target dataset; second, the LLM reasons upon these features, augmented with state-of-the-art components retrieved via Retrieval-Augmented Generation (RAG), to synthesize the architecture into a structured neural network description, which we term the Neural Architecture Description Language (NADL); finally, a compiler instantiates this description into a deployable model. Extensive experiments on five diverse object detection datasets demonstrate that our proposed Cognitive-YOLO consistently generates superior architectures, achieving state-of-the-art (SOTA) performance by outperforming strong baseline models across multiple benchmarks.

## 1 INTRODUCTION

Object detection is a crucial task in computer vision, aimed at identifying and localizing objects within images. The field has rapidly evolved from two-stage approaches like the R-CNNGirshick et al. (2014) family to one-stage methods such as the YOLORedmon et al. (2015) series. Training models on large amounts of data allows them to learn richer features, thereby improving their generalization ability in new scenarios. However, this also brings new challenges. In vertical fields, models need to be fine-tuned on specific datasets, and even improve the structure of general object detection models. A large number of YOLO-based papers are published every year, yet they still fail to meet the needs of more diverse and richer scenes and datasets. Modifying the model structure requires a deep understanding of the dataset, as well as knowledge of various new modules in the object detection task and domain. With the rapid development of the field, new modules and methods continue to emerge, increasing the pressure on researchers, and the continuous modification of various refined scenarios is also time-consuming.

Large language models (LLMs)Hoffmann et al. (2022); Kaplan et al. (2020) have achieved remarkable success in various fields over the past few years. Retrieval-Augmented Generation (RAG) improves model performance by retrieving external knowledge, allowing the model to dynamically access relevant information during inference without requiring retraining. Vision-language models (VLMs) have also shown outstanding performance in grounding tasks, such as GPT-4V(ision)OpenAI (2023a;b;c) and Qwen-VLBai et al. (2025) However, the application of LLMs in computer vision is still in its infancy, and the large computational requirements of VLMs make them difficult to deploy on edge devices.

Neural Architecture Search (NAS) is a technique for automating the search for neural network architectures, aiming to find the optimal network structure through search algorithms. By combining

large language models (LLMs) with RAG, we can leverage their capabilities in understanding and generation to guide the NAS process. However, existing LLM-guided methods primarily focus on optimizing architectures within a given population, treating the LLM as a refinement tool that operates on the architecture space itself. Our work diverges from this paradigm by proposing a ”data-first” approach. We posit that a truly effective architect should reason from the fundamental properties of the problem domain. Consequently, our framework empowers the LLM to act as a holistic architect, synthesizing a high-quality initial architecture by directly reasoning upon the intrinsic characteristics of the dataset—such as object scale distribution, image resolution, and class imbalance. This shifts the role of the LLM from an iterative operator to a primary designer, aiming to generate a robust solution from first principles.

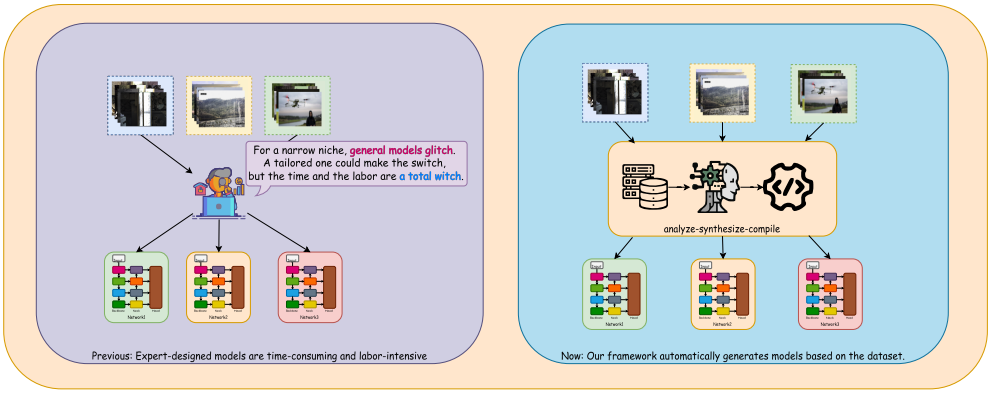


Figure 1: Comparison between past and present model design approaches: General models perform poorly in vertical scenarios. In the past, expert-designed models could improve performance but were time-consuming and labor-intensive. Now our ”analyze-synthesize-compile” Cognitive-YOLO automatically generates models based on data

We propose Cognitive-YOLO, an automatic design method for object detection models based on large language models, aiming to automate the design of network structures that can compete with SOTA models according to the characteristics of scenes and datasets. The contributions of this paper are as follows:

1. We propose a novel paradigm for generating object detection network architectures, based on the first principles of the dataset.
2. We design and implement a flexible and decoupled ”analyze-synthesize-compile” framework.
3. We validate the effectiveness of our method, Cognitive-YOLO, on multiple datasets. Experimental results demonstrate that Cognitive-YOLO can design model architectures with performance comparable to State-of-the-Art (SOTA) models.

## 2 Related Work

### 2.1 Object Detection

It is easy for humans to perform object detection, but machines can only solve this problem after the rise of deep learning. Object detection means identifying objects in a given image and providing their categories and locations. Currently, object detection has been widely applied in various fields, such as rail surface detection, rice leaf diseases detection, and fire detection. Early object detection techniques were based on rules and feature extraction, and they performed poorly on new data. Deep learning-based object detection tasks require training on large datasets to develop the ability to recognize objects. The YOLO family has developed rapidly over the years, from its initial proposal to continuous iterations, with its influence becoming increasingly profound. The subsequent development of YOLO has shown a diversified trend. After YOLOv3, different research teams and communities have contributed multiple versions, such as YOLOv4Bochkovskiy et al.

(2020), YOLOv5ultralytics (a), YOLOv8, YOLOv11ultralytics (b). Furthermore, the Transformer architecture, which initially achieved tremendous success in natural language processing, has also been introduced to computer vision, giving rise to new detection paradigms. For example, the DETR (DEtection TRansformer)Carion et al. (2020) model treats object detection as a set prediction problem. It uses the Transformer's encoder-decoder structure to directly output a non-redundant set of objects, The DEIMHuang et al. (2025) enhances the matching mechanism in DETR by improving the bipartite graph matching process between predicted results and ground truth labels, making it more efficient and stable. This approach can significantly accelerate the model's convergence speed and improve the final detection accuracy without changing the model structure. With the rise of LLMs, open-vocabulary detection methods have also been proposed, such as YOLO-WorldCheng et al. (2024).

### 2.2 LLM AND RAG

In recent years, Large Language Models (LLMs) have achieved breakthrough progress in the field of artificial intelligence and demonstrated powerful natural language understanding and generation capabilities. LLMs are typically based on the Transformer architecture and master rich linguistic patterns and world knowledge through self-supervised learning on massive text data. From early models like Devlin et al. (2018) and GPT-2Radford et al. (2019) to later GPT-3Brown et al., LLaMATouvron et al. (2023); Grattafiori et al. (2024), as well as Qwen, DeepSeekDeepSeek-AI et al. (2024) and other models, both the parameter scale and performance of LLMs have achieved exponential growth. In the past year, reasoning models, such as ChatGPT O1 and DeepSeek R1DeepSeek-AI et al. (2025), have made significant progress in various fields, with long Chain-of-Thought (CoT)Wei et al. (2022) being beneficial for models to perform reasoning and solve complex problems.

The core idea of RAGLewis et al. (2020) is to combine pre-trained language models with external, updatable knowledge bases, thereby integrating the powerful reasoning and generation capabilities of LLMs with the precise and reliable information retrieval capabilities of retrieval systems. When receiving user input (Query), the system first uses this input to retrieve the most relevant information fragments or documents from a large-scale knowledge base. The retrieved relevant information fragments serve as additional context, which is integrated with the user's original question into a prompt. Subsequently, this enhanced prompt is fed into the LLM, guiding it to generate final, well-grounded answers. Agentic searchSingh et al. (2025) is a technique that combines large model agents with retrieval systems, leveraging ReActYao et al. (2023)'s reflection, planning, tool use, and multi-agent collaboration to dynamically manage retrieval strategies and iteratively improve context understanding and query expression.

### 2.3 NEURAL ARCHITECTURE SEARCH

Neural Architecture Search (NAS) has emerged as a key paradigm for automating the design of deep neural networks, aiming to discover optimal architectures for specific tasks without extensive manual intervention. Early approaches, often relying on reinforcement learning or evolutionary algorithms, demonstrated significant success but were hampered by prohibitive computational costs, frequently requiring thousands of GPU-days to find a single high-performance model Lin et al. (2021). This challenge spurred the development of more efficient methods, chief among them being Zero-Cost NAS (ZC-NAS).

The core idea of ZC-NAS is to evaluate and rank candidate architectures without performing any training, thereby drastically reducing search time Lin et al. (2021); Dong et al. (2024). This is achieved through the use of "zero-cost proxies," which are carefully designed metrics that correlate well with a model's final trained performance. Methodologies in this domain have evolved from handcrafted proxies based on network expressivity, such as the Zen-Score Lin et al. (2021), to more advanced frameworks that employ techniques like genetic programming to automatically discover novel proxy metrics from mathematical primitives, as demonstrated by LPZero Dong et al. (2024). These training-free methods provide the foundation for rapid architecture evaluation in modern NAS frameworks.

More recently, the advent of Large Language Models (LLMs) has opened a new frontier for NAS. Researchers have begun to leverage the extensive world knowledge and reasoning capabilities of LLMs to guide the search process. These applications can be broadly categorized into two main

roles. Firstly, LLMs have been successfully employed as powerful performance predictors. In this role, the LLM takes an architecture's description as input and directly estimates its performance on a downstream task, serving as a cost-effective surrogate for the expensive training-and-evaluation cycle Jawahar et al. (2023). Secondly, and more commonly, LLMs are used as intelligent operators within an evolutionary search framework. Works like RZ-NAS and LLMatic utilize LLMs as sophisticated "mutators," generating improved architectural variants based on an existing candidate and its performance score Ji et al. (2025); Nasir et al. (2024). RZ-NAS further enhances this process by incorporating a reflective mechanism, allowing the LLM to learn from past mutations and provide linguistic feedback for future iterations Ji et al. (2025).

It is worth noting that Deci AI also proposed a work named YOLO-NAS Aharon et al. (2021). However, their approach primarily relies on their proprietary AutoNAC neural architecture search technology to discover quantization-friendly architectures optimized for specific hardware, particularly edge devices. In contrast, our Cognitive-YOLO framework does not employ NAS for searching; instead, it positions the LLM as a "holistic architect" that directly reasons upon and synthesizes the architecture from the first principles of the dataset. Thus, despite the previous name similarity (YOLO-NAS), the two works differ fundamentally in core methodology (LLM synthesis vs. NAS discovery) and optimization goals (data-driven vs. hardware-driven).

## 3 METHODOLOGY

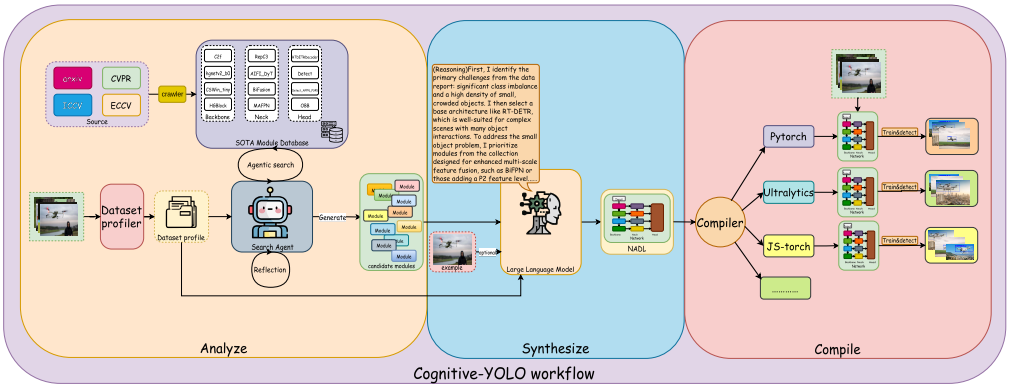


Figure 2: Comparison between past and present model design approaches: General models perform poorly in vertical scenarios. In the past, expert-designed models could improve performance but were time-consuming and labor-intensive. Now our "analyze-synthesize-compile" Cognitive-YOLO automatically generates models based on data

### 3.1 KNOWLEDGE-AUGMENTED SOTA MODULE RETRIEVAL

To ensure that our designed architectures can leverage state-of-the-art technologies, we first construct a dynamically updated knowledge base of SOTA modules. This knowledge base is maintained by an automated web crawler system that periodically scrapes new papers claiming SOTA from preprint archives like arXiv and scans historical proceedings from top-tier conferences. The crawler system attempts to automatically parse the implementation code for modules from the papers and their associated code repositories; modules that cannot be processed automatically are flagged for manual implementation and verification by experts. Each module in the knowledge base is stored in a structured format, containing not only its implementation code but also detailed records of its advantages, disadvantages, scope of application (e.g., effectiveness for small object detection or occluded scenes), and performance metrics.

During the retrieval phase, instead of generic agentic search, we designed a specialized **Data-Driven Architect Agent**. This agent operates on a ReAct (Reason-Act) framework, where its core task is not to perform vague text searches, but to systematically map the dataset's "first principles" to a validated set of "Candidate Modules" using a highly specialized toolset. This design mechanically forces the agent to reason in a data-centric manner.

The agent's first action is to call a tool to "understand" the data. It begins by analyzing the key meta-features generated during the dataset analysis phase, as detailed in Table 1.

Table 1: Key Dataset Meta-features and Their Architectural Implications for the LLM Designer.

| Dimension | Meta-feature | Architectural Implications |
|---|---|---|
| **Overview** | Scale Statistics | Total images and object counts provide a baseline for model capacity selection (e.g., deeper/wider backbone for larger, more complex datasets). |
| **Class Distribution** | Class Imbalance | High imbalance suggests long-tail strategies (e.g., Focal Loss, resampling). |
| | Class Co-occurrence | High co-occurrence favors context-aware modules (e.g., attention, transformers) to model inter-class relationships. |
| **Image Characteristics** | Resolution & Aspect Ratio | High or varied resolutions influence input scaling strategies and the design of the FPN to preserve spatial details. |
| | Color & Texture | Low contrast or simple textures may require a deeper backbone or channel attention (e.g., SE, ECA) for better feature discrimination. |
| **Object Characteristics** | Object Scale (Relative & Absolute) | **Small Object Dominance:** Prioritizes high-res features, advanced FPNs (e.g., BiFPN, AFPN), or specialized small object heads. |
| | Object Aspect Ratio | Extreme ratios suggest using deformation-aware modules (e.g., Deformable Convolutions) or customized anchor designs. |
| **Scene Complexity** | Object Density & Proximity | **Crowded Scenes:** Require advanced NMS algorithms (e.g., Soft-NMS) and loss functions sensitive to precise localization (e.g., SIoU). |
| | Spatial Clustering | High clustering of objects suggests a need for modules enhancing local feature aggregation, like SPPF or RFB. |
| | Spatial Position Bias | Strong positional priors (e.g., objects always on the horizon) can be exploited by position-aware modules like Coordinate Attention. |

To execute this data-driven reasoning, the agent is equipped with the following unique toolset, designed specifically for this task:

- **`get_architectural_drivers(report)`**: A rule-based interpreter tool. It does not simply read the JSON report, but rather "translates" it into a set of high-level, actionable "Architectural Drivers." For example, it maps the raw statistics `"avg_objects_per_image: 1.24"` and `"images_with_no_objects: 7833"` to the single driver: `DRIVER::NEEDS_QUERY_BASED_HEAD_FOR_SPARSITY`.
- **`find_modules_by_driver(driver_key)`**: A highly specialized RAG tool. It queries our SOTA knowledge base for modules that are explicitly tagged as solutions for a specific "Architectural Driver." For instance, inputting `DRIVER::NEEDS_QUERY_BASED_HEAD_FOR_SPARSITY` will return relevant modules like "RTDETRDecoder" or "ASSA_Encoder".
- **`validate_coverage(drivers, modules)`**: This is our core **Reflection** mechanism. This tool checks if the currently proposed list of modules logically addresses *all* drivers identified by the `get_architectural_drivers` tool. It will report `"status: Incomplete"` along with any `"missing_drivers"`, forcing the agent to re-plan and fill the logical gap.
- **`estimate_complexity_and_compatibility(modules, scale)`**: A final "Sanity Check" tool. It verifies technical conflicts between modules (like channel mismatches) and estimates if the combined parameter count is within the target budget (e.g., "n" scale).

The agent's ReAct workflow is thus structured and auditable: (1) It first calls `get_architectural_drivers` to understand the "Why" of the problem. (2) It then iterates through each identified driver, using `find_modules_by_driver` to gather potential solutions (the "What"). (3) Next, it calls `validate_coverage` to reflect and ensure its plan is logically complete. (4) Finally, after passing the `estimate_complexity_and_compatibility` check, it forwards this fully-vetted "Candidate Modules" list to the "holistic architect" LLM in the next stage (Section 3.2).

### 3.2 LLM-BASED ARCHITECTURE SYNTHESIS

In this core stage, the LLM acts as the "holistic architect." Its inputs are two key pieces of information: (1) the dataset analysis report we previously generated, which describes the "first principles" of the data; and (2) the list of SOTA candidate modules filtered by the Agentic Search. We have designed a sophisticated prompting strategy that guides the LLM to perform structural reasoning based on data

characteristics (such as object scale range, scene density, etc.), select the most appropriate modules from the candidate list, and organize them organically into a complete and efficient network topology. The output of the LLM is not direct code, but rather a structured intermediate representation that we define and term the "Neural Architecture Description Language" (NADL). NADL, in JSON format, clearly describes the complete structure of the network, the connection topology between modules, and the specific parameters of each module, serving as a universal "design blueprint" for subsequent code generation.

Taking fire detection as an example, Figure 3 shows a comparison between the architecture generated by Cognitive-YOLO and YOLOv12. To address the sparse scenes and numerous negative samples, Cognitive-YOLO introduces a **Transformer Encoder**. Its object-centric query mechanism is more effective at suppressing background noise and reducing false positives than traditional dense prediction heads. To handle the extreme scale variation, an **RTDETRDecoder** is employed. It uses cross-attention to adaptively fuse multi-scale features, providing a more robust solution for detecting objects of vastly different sizes. The framework retains a lightweight backbone, consistent with the dataset's moderate texture complexity. This allows the parameter budget to be intelligently allocated to the advanced head components, maximizing performance where it is most needed.

### 3.3 Decoupled Architecture Instantiation and Automated Validation

To achieve maximum flexibility and extensibility, we adopt a decoupled frontend-backend design. The LLM acts as the frontend, responsible for abstract design (generating the NADL), while a backend compiler is responsible for the concrete code implementation. Our compiler can take the NADL as input and "compile" it to several different backend platforms. Currently, it supports the generation of PyTorch 'nn.Module' code, '.yaml' configuration files for the Ultralytics framework, and the 'torch.js' format for web-based deployment. Crucially, the compilation process is seamlessly integrated with our Continuous Integration/Continuous Deployment (CI/CD) pipeline. Once a new NADL is generated and successfully compiled, the system automatically triggers a training and testing pipeline to validate the performance of the newly generated architecture on the target datasets, thus achieving a fully automated, closed-loop process from data analysis to performance validation.

## 4 Experiments

### 4.1 Results and Analysis

To comprehensively validate the effectiveness and generalization capability of our proposed Cognitive-YOLO framework, we conducted extensive experiments on five datasets from diverse domains. These datasets include rail surface defect detection in the industrial domain, rice disease detection in agriculture, fire detection for public safety, drone detection for low-altitude security, and student behavior recognition in educational settings. We fairly compared the models generated by Cognitive-YOLO against a series of recognized lightweight SOTA models, including the nano (n) versions of YOLOv5, YOLOv8, YOLOv10, YOLOv11, and YOLOv12. The experimental results are presented in Table 2. As shown by the experimental results, our proposed Cognitive-YOLO demonstrates superior performance across the vast majority of metrics and datasets. In the tasks of Rail Surface Defect and Rice Disease detection, the model generated by Cognitive-YOLO achieved state-of-the-art (SOTA) performance on all five key metrics. Particularly in the rail defect task, the mAP@0.5 metric reached 92.8%, surpassing the best-performing baseline, YOLOv12n, by 2.3%, showcasing its powerful comprehensive detection capabilities.

Notably, Cognitive-YOLO achieves these performance gains while maintaining a relatively efficient parameter count. Although the models generated by Cognitive-YOLO (approx. 5.6M-6.7M parameters) are slightly larger than the nano-scale baselines (approx. 1.8M-3.2M parameters), this moderate increase in parameters yields a disproportionately large return in performance. This proves that Cognitive-YOLO does not improve performance by simply stacking parameters, but rather by intelligently generating more efficient and powerful network structures through a deep understanding of the dataset's characteristics. The only exceptions occurred in the mAP@.5:.95 metric for Fire Detection and the Precision (P) metric for Drone Detection, indicating that while highly-optimized SOTA baselines may still hold an advantage on certain specific metrics, our method is more robust in terms of overall performance.

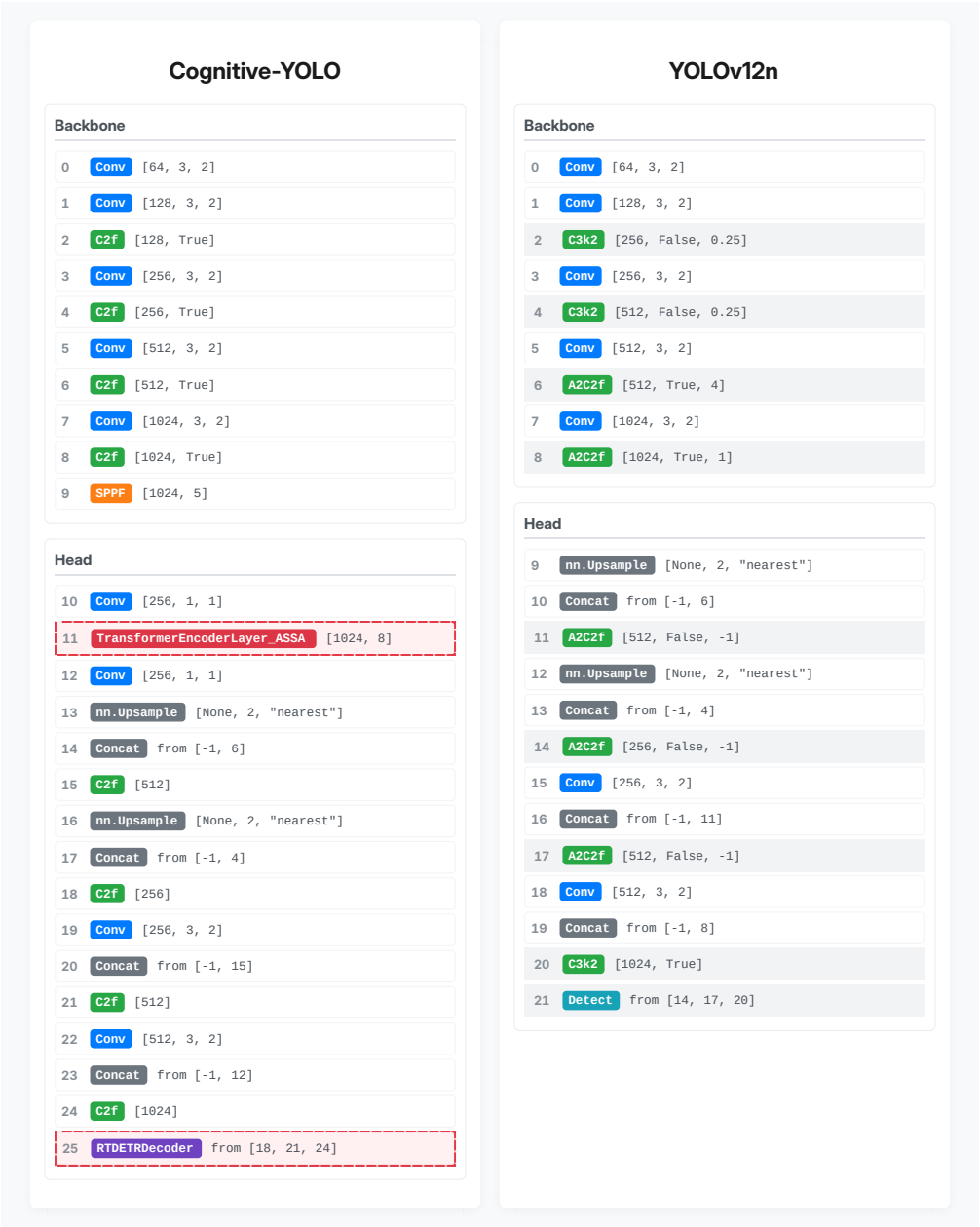


Figure 3: Taking fire detection as an example, compare the differences between the architecture generated by Cognitive-YOLO and YOLOv12.

Table 2: Performance comparison of our Cognitive-YOLO with other SOTA algorithms across five diverse datasets. The best results for each metric within each dataset are highlighted in **bold**.

| Dataset | Algorithm | Params (M) | P (%) | R (%) | mAP@0.5 (%) | mAP@.5:.95 (%) |
|---|---|---|---|---|---|---|
| **Rail Surface Defect** | YOLOv5n | 2.2 | 89.9 | 82.5 | 86.8 | 65.7 |
| | YOLOv8n | 2.7 | 90.3 | 88.7 | 88.4 | 73.3 |
| | YOLOv10n | 2.7 | 88.5 | 86.7 | 89.7 | 68.7 |
| | YOLOv11n | 2.6 | 85.9 | 88.5 | 89.7 | 69.6 |
| | YOLOv12n | 2.5 | 90.0 | 86.6 | 90.5 | 71.6 |
| | **Cognitive-YOLO (Ours)** | 6.7 | **90.6** | **89.6** | **92.8** | **74.3** |
| **Rice Disease** | YOLOv5n | 2.2 | 70.7 | 62.9 | 65.2 | 25.5 |
| | YOLOv8n | 2.7 | 71.2 | 62.1 | 65.0 | 24.3 |
| | YOLOv10n | 2.7 | 69.5 | 65.0 | 64.9 | 23.8 |
| | YOLOv11n | 2.6 | 71.9 | 66.0 | 69.3 | 27.9 |
| | YOLOv12n | 2.5 | 71.1 | 63.0 | 67.1 | 26.2 |
| | **Cognitive-YOLO (Ours)** | 5.9 | **72.9** | **66.4** | **70.4** | **28.0** |
| **Fire Detection** | YOLOv5n | 2.2 | 77.6 | 70.5 | 76.6 | 44.2 |
| | YOLOv8n | 2.7 | 78.1 | 70.2 | 77.3 | 45.6 |
| | YOLOv10n | 2.7 | 78.2 | 70.7 | 77.1 | 45.3 |
| | YOLOv11n | 2.6 | 79.3 | 69.6 | 75.4 | 43.3 |
| | YOLOv12n | 2.5 | 79.9 | 70.4 | 77.4 | **45.9** |
| | **Cognitive-YOLO (Ours)** | 6.2 | **80.2** | **71.7** | **77.6** | 43.2 |
| **Drone Detection** | YOLOv5n | 2.2 | **92.0** | 75.6 | 84.0 | 47.3 |
| | YOLOv8n | 2.7 | 91.8 | 76.6 | 85.3 | 49.7 |
| | YOLOv10n | 2.7 | 85.2 | 76.5 | 80.7 | 46.9 |
| | YOLOv11n | 2.6 | 87.4 | 78.3 | 84.8 | 49.5 |
| | YOLOv12n | 2.5 | 87.9 | 77.7 | 82.1 | 46.1 |
| | **Cognitive-YOLO (Ours)** | 5.6 | 89.2 | **78.4** | **85.8** | **50.6** |
| **Student Behavior** | YOLOv5n | 2.0 | 83.4 | 89.1 | 91.9 | 75.6 |
| | YOLOv8n | 2.5 | 88.7 | 87.6 | 92.9 | 76.9 |
| | YOLOv10n | 2.5 | 88.4 | 84.5 | 92.0 | 76.0 |
| | YOLOv11n | 2.5 | 86.5 | 88.1 | 92.4 | 76.7 |
| | YOLOv12n | 2.5 | 88.0 | 87.3 | 92.7 | 77.5 |
| | **Cognitive-YOLO (Ours)** | 6.1 | **89.7** | **89.3** | **93.0** | **78.0** |

### 4.2 Ablation Studies

To gain a deeper understanding of the contributions of each component within the Cognitive-YOLO framework to the final performance, we conducted systematic ablation studies.

Table 3: Ablation study of Cognitive-YOLO components across five datasets. We report the performance of the full model against two ablated versions: one without the data-driven analysis (*Without Dataset Profile*) and one without the RAG-based module retrieval (*Without RAG*).

| Dataset | Algorithm | Params (M) | P (%) | R (%) | mAP@0.5 (%) | mAP@.5:.95 (%) |
|---|---|---|---|---|---|---|
| **Rail Surface Defect** | Without Dataset Profile | 6.7 | 89.2 | 87.3 | 90.8 | 71.8 |
| | Without RAG | 6.7 | 90.1 | 88.3 | 91.5 | 72.5 |
| | **Cognitive-YOLO (Ours)** | 6.7 | **90.6** | **89.6** | **92.8** | **74.3** |
| **Rice Disease** | Without Dataset Profile | 5.9 | 71.1 | 63.8 | 67.7 | 26.8 |
| | Without RAG | 5.9 | 72.5 | 65.4 | 69.1 | 27.2 |
| | **Cognitive-YOLO (Ours)** | 5.9 | **72.9** | **66.4** | **70.4** | **28.0** |
| **Fire Detection** | Without Dataset Profile | 6.2 | 79.1 | 70.3 | 76.6 | 44.2 |
| | Without RAG | 6.2 | 79.5 | 71.2 | 77.0 | 43.8 |
| | **Cognitive-YOLO (Ours)** | 6.2 | **80.2** | **71.7** | **77.6** | **43.2** |
| **Drone Detection** | Without Dataset Profile | 5.6 | 91.4 | 76.7 | 84.6 | 49.4 |
| | Without RAG | 5.6 | 90.1 | 77.2 | 85.1 | 49.8 |
| | **Cognitive-YOLO (Ours)** | 5.6 | 89.2 | **78.4** | **85.8** | **50.6** |
| **Student Behavior** | Without Dataset Profile | 6.1 | 88.2 | 87.5 | 92.6 | 77.2 |
| | Without RAG | 6.1 | 88.9 | 88.3 | 92.7 | 77.6 |
| | **Cognitive-YOLO (Ours)** | 6.1 | **89.7** | **89.3** | **93.0** | **78.0** |

## 5 CONCLUSION

We proposed and validated Cognitive-YOLO, a novel paradigm that leverages LLMs to automatically design efficient object detection models from the first principles of the dataset. Through our designed decoupled "analyze-synthesize-compile" framework, our method successfully transforms the role of the LLM from an iterative optimizer into an end-to-end holistic architect. The experimental results strongly prove the effectiveness of our approach: on five datasets covering diverse scenarios, the architectures generated by Cognitive-YOLO demonstrated strong competitiveness in both parameter efficiency and detection accuracy, achieving SOTA-level performance. This confirms our core hypothesis that deep data insight is the key driving force for automated and intelligent architecture design.

Despite these encouraging results, our work has several avenues for future exploration. The current knowledge base construction still relies on some expert intervention; future work could explore more fully automated methods for code generation and verification. Furthermore, our framework could be extended to other computer vision tasks, such as instance segmentation and pose estimation, and could also explore the use of more efficient open-source LLMs to reduce inference costs. We believe that this data-driven, LLM-led paradigm for architecture design opens a promising path for the future development of Automated Machine Learning.

## REFERENCES

Shay Aharon, Louis-Dupont, Ofri Masad, Kate Yurkova, Lotem Fridman, Lkdci, Eugene Khvedchenya, Ran Rubin, Natan Bagrov, Borys Tymchenko, Tomer Keren, Alexander Zhilko, and Eran-Deci. Super-gradients, 2021. URL https://zenodo.org/records/7789328.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, February 2025. URL https://arxiv.org/abs/2502.13923.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal Speed and Accuracy of Object Detection. https://arxiv.org/abs/2004.10934, apr 23 2020. [Online; accessed 2025-08-25].

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, May 2020. URL https://arxiv.org/abs/2005.12872.

Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16901–16911. IEEE, June 2024. URL https://doi.org/10.1109/cvpr52733.2024.01599.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, December 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, January 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, October 2018. URL https://arxiv.org/abs/1810.04805.

Peijie Dong, Lujun Li, Xiang Liu, Zhenheng Tang, Xuebo Liu, Qiang Wang, and Xiaowen Chu. LPZero: Language model zero-cost proxy search from zero. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024. arXiv preprint arXiv:2410.04808.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz

Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, g Shen Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ

Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models. https://arxiv.org/abs/2407.21783, jul 31 2024. [Online; accessed 2025-08-25].

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. Deim: Detr with improved matching for fast convergence. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15162–15171. IEEE, June 2025. URL https://doi.org/10.1109/cvpr52734.2025.01412.

Ganesh Jawahar, Muhammad Abdul-Mageed, Laks V. S. Lakshmanan, and Dujian Ding. LLM performance predictors are good initializers for architecture search. *arXiv preprint arXiv:2310.16712*, 2023. Version 2, 7 Aug 2024.

Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. RZ-NAS: Enhancing LLM-guided neural architecture search via reflective zero-cost strategy. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*, 2025. As stated in the paper, placeholder year might be 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 347–356, 2021.

M. U. Nasir, S. Earle, J. Togelius, S. James, and C. Cleghorn. LLMatic: Neural architecture search via large language models and quality-diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24)*, pp. 1110–1118. Association for Computing Machinery, 2024.

OpenAI. ChatGPT can now see, hear, and speak. https://openai.com/blog/chatgpt-can-now-see-hear-and-speak, 2023a. URL https://openai.com/blog/chatgpt-can-now-see-hear-and-speak. Accessed: 2025-08-25.

OpenAI. GPT-4 Technical Report. Technical Report arXiv:2303.08774, OpenAI, 2023b. URL https://arxiv.org/abs/2303.08774.

OpenAI. GPT-4V(ision) System Card. Technical report, OpenAI, 2023c. URL https://cdn.openai.com/papers/GPTV_System_Card.pdf. Accessed: 2025-08-25.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, June 2015. URL https://arxiv.org/abs/1506.02640.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag, 2025. URL https://arxiv.org/abs/2501.09136.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and Efficient Foundation Language Models. https://arxiv.org/abs/2302.13971, feb 27 2023. [Online; accessed 2025-08-25].

ultralytics. Github - ultralytics/yolov5: Yolov5 in PyTorch > ONNX > CoreML > TFLite. https://github.com/ultralytics/yolov5, a. [Online; accessed 2025-08-25].

ultralytics. Github - ultralytics/ultralytics: Ultralytics YOLO. https://github.com/ultralytics/ultralytics, b. [Online; accessed 2025-08-25].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, January 2022. URL https://arxiv.org/abs/2201.11903.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.