

ARID: A Deployable Edge AI System for Structured Information Extraction from Industrial Maintenance Work Orders

Kuanlin Chen
Independent Researcher
Taoyuan, Taiwan
dipper13579@gmail.com
ORCID: 0009-0006-3879-4565

Chen-Wei Kuo
National Tsing Hua University
Hsinchu, Taiwan
ck3294@nyu.edu

Abstract—Maintenance work orders must often be processed offline on embedded hardware, yet downstream software requires predictable structured output. We present ARID (Aviation-inspired Routing for Industrial Deployment), which extracts component, failure mode, symptom, and maintenance action into fixed-schema JSON on an 8GB NVIDIA Jetson Orin NX. ARID combines conservative dual-teacher filtering, targeted noise-aware synthesis, one routing decision per work order, 4-bit inference, and grammar-constrained decoding. From 2,326 unlabeled OMIn records, it retains 716 training pairs and adds 99 topology-constrained records targeting action extraction. On 300 human-labeled records, ARID reaches 84.8% token-F1 on the reference stack and 82.9% on the deployed Jetson. Resident serving achieves 5,310/5,656 ms P50/P99 at 12.5 W. On zero-shot MaintNet transfer, semantic F1 falls to 46.4% while parser success remains at least 99.8%, showing that output validity transfers but field semantics do not.

Index Terms—Industrial Informatics, Edge AI, Maintenance Work Orders, Large Language Models, System Deployment, NVIDIA Jetson

I. INTRODUCTION

Technician narratives contain the evidence needed to populate maintenance databases, but they are short, noisy, and domain-specific. The operational task is therefore stricter than summarization: given one work order, a system must recover four fields—*component*, *failure mode*, *symptom*, and *maintenance action*—and return a record that a CMMS can validate deterministically.

Factory deployment adds three coupled constraints. Logs may be proprietary, the target network may be air-gapped, and an embedded device must provide predictable resident-serving behavior within an 8GB shared-memory budget. At the same time, high-quality field annotations are scarce. Prior maintenance pipelines address extraction or reasoning [1], and prior edge work addresses model execution, but neither result

alone establishes a complete path from noisy text to validated records on the device.

ARID addresses this gap as a hardware-aware pipeline rather than as a new foundation model. Offline, two independent teachers propose structured labels; a conservative agreement filter and targeted Noise-Aware Synthetic Distillation (NASD) construct the fine-tuning set. Online, a semantic gate chooses once per work order whether to apply the domain adapter, a 4-bit model performs extraction, and Grammar-Constrained Decoding (GCD) enforces the JSON grammar. This separation is deliberate: training data affects semantic accuracy, routing determines which specialization is used, and GCD guarantees syntax but cannot repair a wrong field value.

The evaluation makes the same separation. ARID obtains 84.8% token-F1 on the RTX reference stack and retains 82.9% after deployment conversion to llama.cpp Q4_K_M on the Jetson. Resident serving reaches 5,310/5,656 ms P50/P99 at 12.5 W. On MaintNet, parser success remains at least 99.8% while semantic F1 falls to 46.4%, exposing target-domain vocabulary—not JSON formation—as the transfer bottleneck.

The contributions are:

- **A complete measured edge pipeline:** ARID connects low-label training, work-order-level adapter selection, 4-bit inference, and constrained JSON generation on a commercial Jetson Orin NX 8GB.
- **Targeted low-resource data construction:** dual-teacher filtering yields 716 retained pairs, while 99 topology-constrained NASD records produce a significant 1.56 pp action-field gain in the $N = 100$ setting; we distinguish this targeted effect from the nonsignificant overall gain.
- **A semantic-structural transfer diagnosis:** cross-domain evaluation shows that constrained decoding preserves machine validity under shift even when field extraction fails, identifying vocabulary adaptation as the next deployment requirement.

II. RELATED WORK

Industrial maintenance IE. Extraction from maintenance narratives has progressed from statistical classifiers [2] to

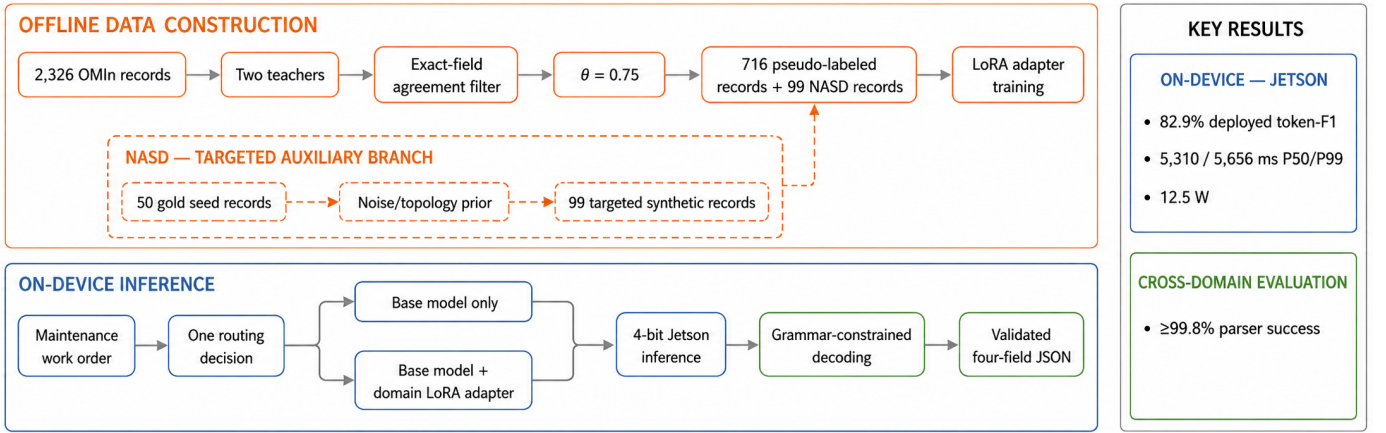


Fig. 1. ARID separates the source of semantic supervision (top) from the deployed extraction path (bottom). The router selects specialization once per record; GCD constrains the JSON grammar but does not determine field content.

resources spanning aviation, automotive, and facility domains. MaintNet [3] provides a collaborative maintenance library; MaintIE [4] defines a fine-grained 224-class schema. OMIn [5] contains FAA-derived incident records and supports knowledge-extraction pipelines such as KEO [1]. These works establish the semantic task; ARID asks what is required to execute a smaller four-field schema locally and return validated records under an embedded-device budget.

Edge LLM deployment. Four-bit quantization reduces the dense-weight footprint of small LLMs [6], [7] and enables execution on constrained devices [8]. Jetson studies characterize model-scale, latency, and power trade-offs [8]; EdgeLoRA [9] and CLONE [10] address multi-adapter serving and latency-aware scheduling. ARID builds on these deployment mechanisms but evaluates an end-to-end maintenance extractor, including output validity and cross-domain behavior.

Mixture-of-LoRA adapter routing. LoRA [11] has been extended to token-level mixtures such as MoLE [12], MoLoRA [13], and LD-MoLE [14]; EdgeMoE [15] separates hot backbone weights from cold expert weights. ARID does not route tokens or compose experts. It makes one auditable base-versus-adapter decision before generation, trading routing expressiveness for a simpler edge execution path.

Structured decoding and annotation bootstrapping. GCD masks illegal tokens according to a finite-state grammar [16]; it has improved structured IE in low-resource settings [17]. Separately, knowledge-graph-grounded prompting can generate realistic maintenance orders [18], and Noise-Aware Training improves robustness to corrupted text [19]. ARID combines these ideas but assigns them different roles: synthesis targets semantic coverage, whereas GCD enforces structure only.

III. ARID SYSTEM DESIGN

Fig. 1 separates offline data construction from the on-device path. A conventional 3B model requires roughly 6GB for FP16 weights before KV cache and runtime buffers; ARID therefore uses a 4-bit backbone and a compact LoRA adapter.

The output contract is a four-key JSON object. The design goal is not to make every field correct by construction—a grammar cannot do that—but to make semantic errors measurable and syntactic failures preventable.

A. Agreement-Filtered Pseudo-Labels

Two independent teachers, Gemini-2.5-Pro and DeepSeek-Chat, map each of 2,326 unlabeled OMIn work orders to the target schema. For record i , exact field agreement is

$$a_i = \frac{1}{4} \sum_{f=1}^4 \mathbb{1}[y_{if}^{(1)} = y_{if}^{(2)}]. \quad (1)$$

For each threshold, records with $a_i \geq \theta$ are retained and fieldwise Cohen’s κ is computed over that retained set. Fig. 2 shows the resulting agreement–retention curve. Kneedle [20] identifies $\theta=0.50$ as the curve knee under a 40% retention floor. We use this knee as a diagnostic, then choose the stricter $\theta=0.75$ because the deployment objective prioritizes label precision over corpus size. This requires exact teacher agreement on at least three of four fields and retains 716 records. The retained subset has fieldwise Cohen’s $\kappa = 0.744$. It is a conservative engineering override, not an automatically chosen optimum. The 40% floor constrains the automatic knee search only; the selected $\theta=0.75$ retains 30.8% and therefore lies below it by design. The seed, pseudo-label, and test pools are disjoint.

B. Noise-Aware Synthetic Distillation (NASD)

The pseudo-labeled set underrepresents the surface diversity of *maintenance action*. NASD is therefore a targeted augmentation, not a replacement for pseudo-labeling. From a disjoint seed of 50 gold records, it derives a topology prior over components, actions, abbreviations, phoneme-confusion typos, and telegraphic grammar. A generator then produces 99 topology-constrained work orders with explicit action nodes. These records are added to the 716 pseudo-labels for adapter training. Section IV-C evaluates the intended action-field effect separately from overall F1.

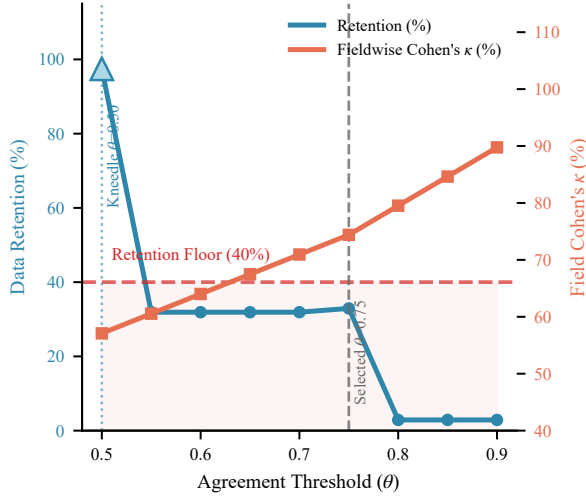


Fig. 2. Teacher-agreement filtering. Kneedle locates the diagnostic knee at $\theta=0.50$; the stricter record-agreement threshold $\theta=0.75$ retains 716 records.

C. Work-Order-Level Adapter Selection

The input is embedded with all-MiniLM-L6-v2 and compared with precomputed general and expert prototypes. If expert similarity exceeds the fixed development-set threshold $\tau = 0.25$, ARID applies the domain LoRA adapter; otherwise it uses the shared base model. The decision is made once before generation, so it adds no per-token routing or adapter-composition loop.

At the selected operating point, 97.3% of OMin test records use the expert path. Consequently, the current router should be interpreted as an auditable specialization policy, not as evidence of large average compute savings on this domain. In resident serving, the adapter remains in memory; the cold swap experiment in Section IV-D quantifies the alternative disk-load path.

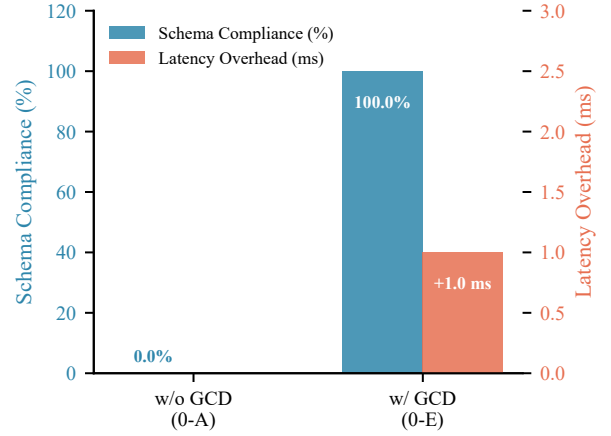
D. Grammar-Constrained Decoding (GCD)

The four-field schema is compiled into a finite-state machine that masks tokens violating the JSON grammar. GCD therefore guarantees character-level grammar validity, while a deterministic parser verifies that all required keys are present. The distinction matters: GCD prevents malformed payloads but cannot correct a semantically wrong component or action. In the on-device check ($N = 10$), its median overhead is 1 ms; across the full evaluations, grammar validity is 100% and unseen-domain parser success is at least 99.8%.

IV. EXPERIMENTS AND HARDWARE VALIDATION

A. Experimental Setup

Evaluation backends. The reference stack uses bitsandbytes 4-bit quantization and outlines 1.2.12 on an RTX 3060 Ti (12 GB). The deployed stack uses llama.cpp Q4_K_M on an Aetina AIB-SN31 with a Jetson Orin NX 8GB, JetPack 6.2, and the 20 W power mode. We report the backend beside every accuracy value and reserve P50/P99 latency and power claims for measurements on the Jetson.



Latency measured on Jetson Orin NX 8GB (AIB-SN31), $N=10$

Fig. 3. GCD on the Jetson Orin NX 8GB: 100% character-level JSON grammar validity at 1 ms median overhead; unconstrained zero-shot schema compliance is 0% ($N = 10$).

Model and training configuration. All configurations use Qwen2.5-3B as the backbone. Adapters are trained with QLoRA (4-bit NF4 quantization, double quantization, bfloat16 compute), LoRA rank 8, $\alpha = 16$, dropout 0.05, applied to `q_proj` and `v_proj`. Training uses 3 epochs, effective batch size 16 (per-device 4 with gradient accumulation 4), learning rate 2×10^{-4} , and the AdamW optimizer. The same recipe is used for every adapter reported in this paper.

The in-domain test set contains 300 human-annotated OMin records. Token-F1 is micro-averaged over tokens in the four output fields. *Grammar validity* means that the character sequence obeys the compiled grammar; *parser success* additionally requires a JSON object with all four keys. The routing threshold is fixed at $\tau = 0.25$ from the development sweep and is not returned on either test domain.

B. Main Results

Table I presents the accuracy chain without mixing reference and deployment backends in one cell. Pseudo-labeling supplies the largest training-data gain: 84.7% versus 77.5% for plain synthetic SFT. Adding NASD changes full-data F1 by only 0.1 pp; its narrower low-resource action-field effect is tested next. Conversion from the RTX reference stack to the deployed Jetson stack costs 1.9 pp, yielding 82.9% token-F1 on device.

C. Low-Resource Ablation

To isolate NASD’s intended effect, both low-resource models use the same $N = 100$ pseudo-labeled subset; one additionally receives the 99 NASD records. The overall +0.4 pp change is not significant (Wilcoxon $p = 0.086$), whereas action extraction improves by 1.56 pp (McNemar $p < 0.001$). Thus NASD is supported as a targeted action-field intervention under scarcity, not as the source of ARID’s overall 84.8% result.

TABLE I
OMIN EXTRACTION ACCURACY ($N = 300$). BACKEND IS SHOWN EXPLICITLY.

Configuration	Backend	Token-F1	Grammar
Zero-shot, no GCD	RTX	0.0% [†]	0%
Plain synthetic SFT	RTX	77.5%	100%
Pseudo-label SFT	RTX	84.7%	100%
+ NASD (ARID reference)	RTX	84.8%	100%
ARID deployed	Jetson	82.9%	100%
Cloud API zero-shot	Cloud	67.3%	100% [¶]

RTX: bitsandbytes 4-bit + outlines. Jetson: llama.cpp Q4_K_M. The deployed row uses the fixed router ($\tau = 0.25$, expert activation 97.3%).

[†]Without fine-tuning or GCD, the backbone produced no parseable four-field JSON; all fields were therefore scored as missing.

[¶]OpenAI GPT-5 Nano (gpt-5-nano-2025-08-07), evaluated via API on March 21, 2026. Schema compliance comes from prompt-based instruction following; no GCD was applied. This row is a deployment-contrast reference, not a same-supervision upper bound. External API latency is omitted because connectivity-dependent estimates are not commensurate with on-device measurements.

TABLE II
LOW-RESOURCE ABLATION: NASD VS. PSEUDO-LABEL ONLY ($N = 100$)

Training Data	Token-F1	Action F1
100 pseudo only	81.2%	26.9%
99 NASD + 100 pseudo	81.6%	28.5%
Difference	+0.4 pp [§]	+1.56 pp ^{**}

[§]Not significant (Wilcoxon $p = 0.086$).

^{**}Significant (McNemar $p < 0.001$).

Wilcoxon tests are conducted over per-sample token-F1 distributions; McNemar tests are applied to paired binary correctness at the field level.

D. Latency and Hardware Validation

All latency measurements use batch size 1, greedy decoding (temperature = 0), and a fixed output budget of 80 tokens. For resident serving, model weights, adapter, and grammar compiler remain in memory via llama-server. $N = 300$ test samples are evaluated sequentially.

Table III reports only measured Jetson configurations. External API latency is omitted because connectivity-dependent estimates are not commensurate with on-device measurements.

Fig. 4 shows why residency matters: loading the adapter from disk adds 1,495 ms to the median path and broadens the tail. ARID therefore uses llama-server with the backbone, adapter, and grammar resident for normal operation; cold swap is retained only as a diagnostic configuration.

During sustained inference, the device draws 12.5 W without thermal throttling (Fig. 5). Together with the measured latency distribution, this establishes a reproducible resident-serving operating point; it is not a hard real-time worst-case guarantee.

E. Domain Generalization

ARID is applied zero-shot to MaintNet [3], whose ground-equipment and facility vocabulary differs from OMin aviation records. Overall token-F1 drops from 84.8% to 46.4% (Fig. 6). Component extraction remains 82.0% (−18 pp), while symptom extraction collapses to 12.6% (−85 pp). Yet end-to-end parser success remains at least 99.8%. The contrast isolates

TABLE III
MEASURED JETSON LATENCY ($N = 300$, BATCH SIZE 1).

Execution path	P50	P99	Disk-load cost
Resident serving	5,310 ms	5,656 ms	0 ms
Cold swap diagnostic	6,805 ms	11,481 ms	1,495 ms

Jetson Orin NX 8GB, llama.cpp Q4_K_M, 20 W mode. Resident P50/P99 exclude the first model load; observed maximum after that exclusion is 11,865 ms.

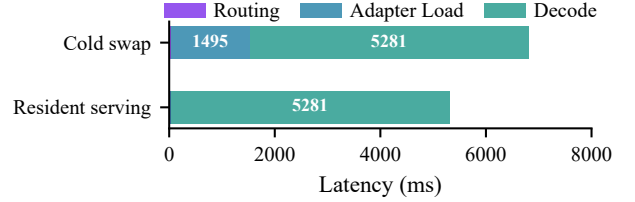


Fig. 4. Inference latency breakdown on the Jetson Orin NX 8GB (AIB-SN31, 20 W mode). Resident serving retains the adapter in memory via llama-server; cold swap reloads it from disk via llama-cli, adding 1,495 ms.

two forms of portability: the output contract transfers because it is enforced at decoding time; field semantics do not transfer without target-domain vocabulary adaptation.

F. Scope of the Claims

Three results bound the contribution. First, the 97.3% expert activation rate means the router provides explicit specialization control on OMin, not sparse-expert efficiency. Second, NASD’s full-data gain is 0.1 pp; only the low-resource action-field effect is supported statistically. Third, 46.4% MaintNet F1 is not production-ready cross-domain accuracy. ARID establishes an offline execution and validation path; a new maintenance domain still requires semantic adaptation.

V. CONCLUSION

ARID turns four-field maintenance extraction into a measured offline pipeline on an 8 GB Jetson: 82.9% deployed token-F1, 5,310/5,656 ms resident P50/P99, 12.5 W active power, and grammar-valid JSON. Its most useful finding is also its boundary: constrained decoding preserves the machine interface under domain shift, but it does not preserve semantic accuracy. Reliable deployment therefore requires both a stable edge execution path and target-domain adaptation; ARID provides the former and makes the latter visible.

ACKNOWLEDGMENT

OpenAI IMAGE-2 was used to generate the visual draft of Fig. 1 from author-written prompts; the authors selected, corrected, and verified the final diagram. OpenAI Codex was used for language editing and consistency checks throughout the manuscript. The authors reviewed all AI-assisted output and remain responsible for the content.

REFERENCES

- [1] K. Ai, J. A. Karr, Jr., M. Jiang, N. V. Chawla, and C. Wang, “KEO: Knowledge extraction on OMin via knowledge graphs and RAG for safety-critical aviation maintenance,” *arXiv:2510.05524*, 2025.

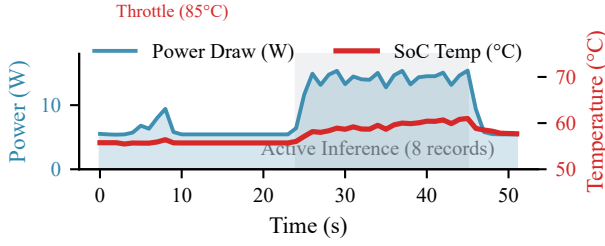


Fig. 5. SoC power draw and temperature on the Aetina AIB-SN31 (20 W mode). Active inference draws 12.5 W; the system remains within thermal limits without throttling.

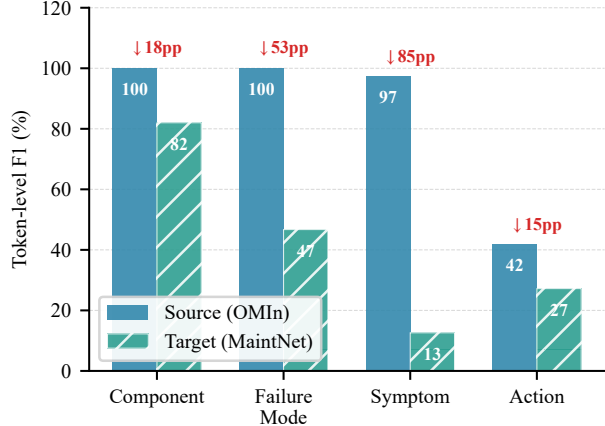


Fig. 6. Per-field token-F1 on OMIn and MaintNet. *Component* is more domain-generic; *Symptom* exposes the vocabulary bottleneck. Parser success remains $\geq 99.8\%$ under GCD.

- [2] U. Rahman and M. U. Mahbub, “Application of classification models on maintenance records through text mining approach in industrial environment,” *J. Qual. Maint. Eng.*, vol. 29, no. 1, pp. 203–219, 2023.
- [3] F. Akhbardeh, T. Desell, and M. Zampieri, “MaintNet: A collaborative open-source library for predictive maintenance language resources,” in *Proc. 28th Int. Conf. Computational Linguistics: System Demonstrations*, 2020, pp. 7–11.
- [4] T. K. Bikaun, T. French, M. Stewart, W. Liu, and M. Hodkiewicz, “MaintNet: A fine-grained annotation schema and benchmark for information extraction from maintenance short texts,” in *Proc. LREC-COLING*, 2024, pp. 10939–10951.
- [5] K. P. Mealey, J. A. Karr, Jr., P. Saboia Moreira, P. R. Brenner, and C. F. Vardeman, II, “Trusted knowledge extraction for operations and maintenance intelligence,” *Nat. Lang. Process. J.*, vol. 13, p. 100187, 2025.
- [6] L. Sparrenberg, T. Deußer, A. Berger, and R. Sifa, “Small and fast LLMs on commodity hardware: Post-training quantization in llama.cpp,” in *Proc. IEEE DSAA*, 2025, pp. 1–10.

- [7] U. Kurt, “Which quantization should i use? A unified evaluation of llama.cpp quantization on llama-3.1-8b-instruct,” *arXiv:2601.14277*, 2026.
- [8] M. Arya and Y. Simmhan, “Understanding the performance and power of LLM inferencing on edge accelerators,” in *Proc. IEEE Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)*, 2025, pp. 1108–1111.
- [9] Z. Shen, Y. He, Z. Wang, Y. Zhang, G. Sun, W. Ye, and A. Li, “EdgeLoRA: An efficient multi-tenant LLM serving system on edge devices,” in *Proc. ACM MobiSys*, 2025, pp. 124–137.
- [10] C. Tian, X. Qin, K. Tam, L. Li, Z. Wang, Y. Zhao, M. Zhang, and C. Xu, “CLONE: Customizing LLMs for efficient latency-aware inference at the edge,” in *Proc. USENIX ATC*, 2025, pp. 563–585.
- [11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-rank adaptation of large language models,” in *Proc. ICLR*, 2022.
- [12] X. Wu, S. Huang, and F. Wei, “Mixture of LoRA experts,” in *Proc. ICLR*, 2024.
- [13] S. Shah and J. Wagle, “MoLoRA: Composible specialization via per-token adapter routing,” *arXiv:2603.15965*, 2026.
- [14] Y. Zhuang, Y. Shen, Y. Bian, Q. Su, S. Ji, Y. Shi, and F. Miao, “LD-MoLE: Learnable dynamic routing for mixture of LoRA experts,” in *Proc. ICLR*, 2026.
- [15] R. Yi, L. Guo, S. Wei, A. Zhou, S. Wang, and M. Xu, “EdgeMoE: Empowering sparse large language models on mobile devices,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 8, pp. 7059–7073, 2025.
- [16] B. T. Willard and R. Louf, “Efficient guided generation for large language models,” *arXiv:2307.09702*, 2023.
- [17] D. M. Schmidt and P. Cimiano, “Grammar-constrained decoding for structured information extraction with fine-tuned generative models applied to clinical trial abstracts,” *Front. Artif. Intell.*, vol. 7, p. 1406857, 2025.
- [18] A. Lau, J. Feng, M. Hodkiewicz, C. Woods, M. Stewart, and A. Polpo, “Generating authentic grounded synthetic maintenance work orders,” *IEEE Access*, vol. 13, pp. 145 888–145 904, 2025.
- [19] M. Namysl, S. Behnke, and J. Köhler, “NAT: Noise-aware training for robust neural sequence labeling,” in *Proc. ACL*, 2020, pp. 1501–1517.
- [20] V. Satopää, J. Albrecht, D. Irwin, and B. Raghavan, “Finding a “kneedle” in a haystack: Detecting knee points in system behavior,” in *Proc. ICDCS Workshops*, 2011, pp. 166–171.