symmetry

MDPI

Article

Complexity Analysis for Categorized Edge Language Models

Niks Kordjukovs and Danilo Pietro Pau *

System Research and Applications, STMicroelectronics, 20864 Agrate Brianza, Italy; niks.kordjukovs@st.com
* Correspondence: danilo.pau@st.com

Abstract

Edge generative artificial intelligence (AI) increasingly combines language, perception, reasoning, audio, and action on resource-constrained devices. This paper profiles public GPT-Generated Unified Format (GGUF) checkpoints from the Hugging Face Hub (HFH) across conversational, instruct, thinking, audio, vision-language (VL), and vision-language-action (VLA) categories using a shared parser-based deployment-envelope workflow. The main category-specific run retained 21,039 profiled entries and estimated the minimum memory bandwidth, compute throughput, and unified-memory architecture (UMA) footprint needed to satisfy category-specific target throughput values. The resulting measurement protocol was symmetric, but the deployment envelopes were asymmetric: VL and thinking workloads were the heaviest on the compute–bandwidth axis, VLA formed a smaller elevated multimodal branch, and audio, instruct, and conversational workloads were lighter on average. A unified 10-tokens-per-second (TPS) sensitivity run compressed the compute–bandwidth gaps, showing that service-rate assumptions contributed strongly to cross-category separation. Welch/Games–Howell and Kruskal/Dunn analyses confirmed large category effects for bandwidth and compute in the category-specific regime, but only small memory effects. The results show that edge-model feasibility cannot be inferred from parameter count alone; throughput target, backbone family, modality, and memory budgeting must be considered jointly.

Keywords: symmetry/asymmetry; edge AI; GGUF; memory bandwidth; multimodal inference; workload complexity; UMA; backbone heterogeneity

1. Introduction

Edge AI has moved from primarily text-centric assistants toward heterogeneous workloads that combine language, perception, and action, especially on mobile, embedded, and robotic platforms [1–3]. On such platforms, deployment feasibility is constrained not only by model size, but also by memory bandwidth, compute throughput, and available memory [4–6]. In that context, downstream capability alone is insufficient to characterize feasibility, and deployment-oriented complexity profiling is needed alongside task accuracy. The target deployment abstraction adopted here was a simple UMA, namely a central compute element connected to RAM through a shared bus.

The motivation has become more pressing as recent VL and VLA systems such as Helix, OpenVLA, and SmolVLA have shown that multimodal perception, language understanding, and action generation are increasingly being pushed toward generalist embodied operation [7–9]. That trend increased the importance of format-level compatibility, parser-level observability, and hardware-envelope estimation for models intended to run at the edge.

Received: 21 March 2026
Revised: 23 April 2026
Accepted: 25 April 2026
Published: 29 April 2026

Despite rapid progress in edge inference, most prior studies benchmark a limited set of models on fixed hardware, focus on a single workload class, or prioritize runtime optimization rather than cross-category deployment comparison [3–5]. A gap therefore remains in platform-independent profiling that compares minimum feasible hardware envelopes across heterogeneous generative workloads under one measurement protocol. The present study addressed that gap by analyzing public GGUF checkpoints from HFH across six workload categories relevant to edge deployment: conversational, instruct, thinking, audio, VL, and VLA models [10]. A common profiling pipeline was used to discover heterogeneous repositories, normalize parser outputs, and estimate the minimum hardware envelope required to satisfy fixed throughput targets. The paper was framed around a symmetry–asymmetry question: under a symmetric measurement protocol, which deployment costs remained comparable across categories and which diverged sharply? In this study, the symmetry lay in the shared discovery, parsing, threshold search, and summary rules, whereas the asymmetry emerged in the resulting compute, bandwidth, memory, and backbone distributions.

The analysis was guided by four research objectives. First, it quantified the minimum bandwidth, compute, and UMA memory envelopes associated with each workload category under deployment-relevant target generation rates. Second, it tested whether the observed category differences persisted in the compact subset below 4 B parameters. Third, it separated category effects from service-rate assumptions by comparing the category-specific regime with a unified 10-TPS sensitivity run. Fourth, it examined how backbone composition, auxiliary GGUF packaging, and context-window choices affected the interpretation of the resulting deployment envelopes.

Across all categories, the retained corpus contained two artifact-layout cases. A backbone-only repository exposed the main GGUF checkpoint used for parser estimation without a detected companion projection artifact. An auxiliary-GGUF repository exposed the main GGUF checkpoint together with one or more companion GGUF components, most commonly multimodal projection files such as `mmproj`. The auxiliary label therefore described repository packaging rather than a separate workload category. Restricting the corpus to GGUF checkpoints was deliberate: it held the artifact format constant across categories, exposed comparable metadata for parser-based inspection, and aligned the study with widely used edge deployment tooling such as `llama.cpp` and `gguf-parser-go` [11,12].

Table 1 clarified the operational scope of the six categories and, in particular, made explicit that the conversational category referred to multi-turn dialogue workloads rather than to generic text generation. The distinction was important for interpreting later resource comparisons because the instruct and thinking categories also produced text but were grouped by different task structure: instruct models emphasized prompt-to-task execution, whereas thinking models emphasized multi-step reasoning. The table also separated VL from VLA by whether an action output was part of the workload, and it restricted the audio category to token-generating audio-language systems that remained compatible with the GGUF parsing pipeline used in this study.

These category boundaries were defined by deployment-facing interaction modes rather than by benchmark labels alone. Conversational and instruct were separated because a persistent dialogue state and prompt-to-task execution stress the edge system differently; thinking was isolated because reasoning-oriented generation can inflate internal token production and therefore latency pressure; audio was limited to token-generating audio-language checkpoints because streaming speech workloads impose synchronization constraints that differ from text-only decoding; and VL was separated from VLA because producing text and producing control actions correspond to different real-world endpoints, from document understanding to embodied robotics [3,7,8,13]. This framing tied each

class to a recognizable deployment setting: chat assistants, offline AI assistants , speech interfaces, multimodal perception, and robot control—so that later hardware comparisons remained operationally interpretable.

Table 1. Operational definitions of the six workload categories and representative task families used throughout the study.

Category	Operational Definition	Representative Task Families
Conversational	Multi-turn dialogue models optimized to maintain context across successive user–assistant turns and to produce chat-oriented responses.	Open-ended chat, dialogue continuation, follow-up question answering, role-conditioned assistance, and assistant-style interaction.
Instruct	Primarily instruction-following text models optimized for direct task execution from a prompt rather than sustained dialogue state.	Single-turn question answering, summarization, extraction, transformation, classification, and structured text generation.
Thinking	Reasoning-oriented models tuned to externalize or preserve multi-step deliberation before producing a final answer.	Mathematical reasoning, logical inference, planning, puzzle solving, and long-form chain-of-thought style problem decomposition.
Audio	Token-generating audio-language models that accepted audio or speech inputs within the GGUF-compatible tokenized inference stack.	Spoken question answering, audio-conditioned dialogue, speech-grounded instruction following, audio captioning, and transcription-adjacent multimodal assistance.
VL	Vision-language models that combined visual inputs with text generation but did not emit control actions.	Image question answering, captioning, optical character recognition (OCR)-assisted understanding, document interpretation, and multimodal chat.
VLA	Vision-language-action models that extended multimodal understanding to action prediction or embodied control outputs.	Robot command following, perception-conditioned action generation, embodied planning, manipulation-oriented instruction following, and generalist control policies.

The main contributions of the study were fourfold:

- A large-scale public corpus of 21,039 profiled GGUF checkpoints spanning six deployment-relevant workload categories;
- A symmetric discovery, parser normalization, and bandwidth-first threshold-search workflow for estimating minimum feasible gigabytes per second (GBps), giga floating-point operations per second (GFLOPS), and UMA requirements;
- An explicit operational framing of workload classes that linked category boundaries to real-world edge deployment scenarios;
- A cross-category analysis showing that workload class, service-rate target, and backbone family jointly shaped the resulting deployment envelopes more strongly than parameter count alone.

The remainder of this paper is organized as follows. Section 3 describes the toolchain, corpus-construction pipeline, parser-normalization stage, bandwidth-first threshold optimization routine, and inferential-statistics module. Section 4 reports per-category statistics, graph-based operating envelopes, backbone-unique frontier tables, and cross-category comparisons. Section 5 interprets the resulting symmetry–asymmetry pattern and summarizes the main limitations. Section 6 concludes the paper.

2. State of the Art

Peer-reviewed studies have increasingly addressed efficient inference in large language models on edge and mobile hardware from both survey and empirical perspectives. Recent review articles in *ACM Computing Surveys*, *Computer Science Review*, and *Tsinghua Science and Technology* synthesized the design, execution, and optimization challenges of

edge deployment, emphasizing memory limits, bandwidth pressure, quantization, and heterogeneous hardware constraints [1,2,14]. Complementing these surveys, empirical studies such as the ACM Southeast analysis of deployment footprints and the MobiCom study MELTing Point evaluated the practical latency, memory, and feasibility constraints of running language models on edge-class systems [4,5].

A second closely related strand examined architectural and systems-level efficiency. SparQ Attention showed that inference can often be improved by reducing memory traffic rather than by increasing arithmetic throughput alone, which directly supports a bandwidth-aware deployment perspective [6]. For multimodal generation, Lee et al. demonstrated that resource bottlenecks extend beyond text-only decoding and that multimodal inference introduces additional latency and systems constraints that must be characterized explicitly [3].

At the tooling level, practical ecosystems such as GGUF on the HFH, `llama.cpp`, `gguf-parser-go`, and `hf-agents` have made quantized local deployment, metadata inspection, remote artifact parsing, and hardware-aware orchestration more accessible [11,12,15]. However, these tools are primarily designed for execution, inspection, or orchestration rather than for corpus-scale comparative profiling across workload categories.

Despite this progress, a gap remains between runtime benchmarking on fixed devices and large-scale deployment-envelope analysis across heterogeneous model families. Existing studies typically benchmark a limited set of models on predefined hardware, focus on one workload class, or prioritize runtime optimization rather than symmetric cross-category comparison. In contrast, the present study profiles a large public GGUF corpus and compares minimum feasible deployment envelopes across conversational, instruct, thinking, audio, VL and VLA workloads under a unified bandwidth-first optimization protocol.

Not all sources cited in this manuscript were peer reviewed. Several references correspond to software repositories, documentation pages, or recent preprints that were included only when no archival peer-reviewed counterpart was available at the time of writing.

3. Methodology

3.1. Essential Tools and Formats

Two implementation choices were foundational to the pipeline. First, the corpus was intentionally restricted to the GGUF binary format. This was a methodological control variable rather than a claim that GGUF is the only relevant deployment format. The study required a corpus-scale comparison in which artifact structure, metadata availability, quantization labeling, and parser access were held as constant as possible across categories. GGUF was suitable for that role because it is a self-describing binary format designed for GGML-based executors, with metadata stored through an extensible key–value structure and with the information needed to load the model contained in the artifact itself. These properties reduced dependence on repository naming conventions, framework-specific loaders, and inconsistent external annotations when architecture, quantization, and tensor-related information had to be recovered for large numbers of checkpoints. The official GGUF specification described the format as self-contained, extensible, `mmap`-compatible, and fully informative for loading, while the HFH documentation showed that GGUF artifacts exposed both standardized metadata and tensor information for inspection [10,16].

This restriction necessarily introduced a format-availability bias. Models distributed only as framework-native weights, hosted only behind application programming interfaces, released without parser-readable metadata, or not converted to GGUF were outside the sampling frame. The retained corpus therefore represents the public, GGUF-converted, edge-oriented portion of the model ecosystem rather than the full population of generative

models. This bias could overrepresent model families that have active local-inference communities, permissive distribution terms, smaller or quantization-ready checkpoints, and mature `llama.cpp`/GGML support; it could underrepresent proprietary systems, very large server-oriented models, device-specific vendor runtimes, and recent checkpoints that had not yet been converted to GGUF. Consequently, the absolute category counts and backbone frequencies should not be interpreted as market shares or as evidence that non-GGUF models would have the same deployment envelopes. The design choice does, however, keep the internal comparisons well controlled: all retained rows were evaluated through the same artifact format, parser interface, and manuscript-facing Q4-K-M (4-bit K-block medium) quantization filter.

Second, `gguf-parser-go` was adopted as the resource-estimation backend because it could inspect GGUF checkpoints available through local or remote artifact references, including by reading metadata without transferring the entire checkpoint. In the present study, the tool was used as a profiling and estimation backend rather than as a direct inference engine. This distinction was important. According to its documentation, `gguf-parser-go` was designed to estimate memory usage and, experimentally, maximum TPS from supplied device metrics without running the model, and its reported memory usage usually deviated from actual usage by about 100 mebibytes (MiB). The same documentation also stated that maximum-TPS estimation was experimental. As a result, the tool was suitable for deployment-envelope screening under a shared protocol, but not as a substitute for direct board-level benchmarking. In methodological terms, GGUF and `gguf-parser-go` together provided a common artifact format and a common estimator interface for cross-category complexity comparison [10–12,16].

A further practical reason for this toolchain was that a subset of repositories in all six workload categories separated modality-specific encoders, decoders, or projection blocks into companion GGUF modules. In this paper, backbone-only denotes repositories where the selected artifact was the main model checkpoint and no auxiliary projection GGUF was detected, whereas auxiliary-GGUF denotes repositories that also contained companion GGUF components, most commonly `mmproj` files used by multimodal models. Because `gguf-parser-go` estimated resources from the GGUF artifact that it parsed, and because those companion projections were not represented uniformly across repositories, the estimation stage was anchored to the selected backbone GGUF. The reported GBps, GFLOPS, and UMA values should therefore be interpreted as backbone-side estimates that were kept comparable across conversational, instruct, thinking, audio, VL, and VLA checkpoints. This restriction improved cross-category consistency, but it also meant that the reported values did not represent the full end-to-end multimodal system cost when auxiliary projection components were present.

3.2. Model Discovery and Corpus Construction

A multi-stage automated pipeline was implemented to construct the model corpus and to estimate deployment-oriented complexity metrics for GGUF checkpoints. The workflow consisted of five consecutive stages: candidate discovery on the HFH, workload classification and filtering, normalization of parser outputs into structured tables, two-stage minimization of the hardware resources required to satisfy a predefined throughput constraint, and a final inferential-statistics stage applied to the exported metric tables. The software stack supported the six workload categories discussed in this paper, namely conversational, instruct, thinking, audio, VL, and VLA.

Figure 1 summarizes the end-to-end workflow from repository discovery to the final cross-category comparisons. Primary-category assignment followed the operational definitions in Table 1. The goal was not to claim that model families are ontologically dis-

joint, but to assign each repository to the deployment mode that most directly determines expected latency, bandwidth, and memory requirements. This avoided double counting the same family under multiple service modes and kept downstream comparisons tied to concrete edge use cases.

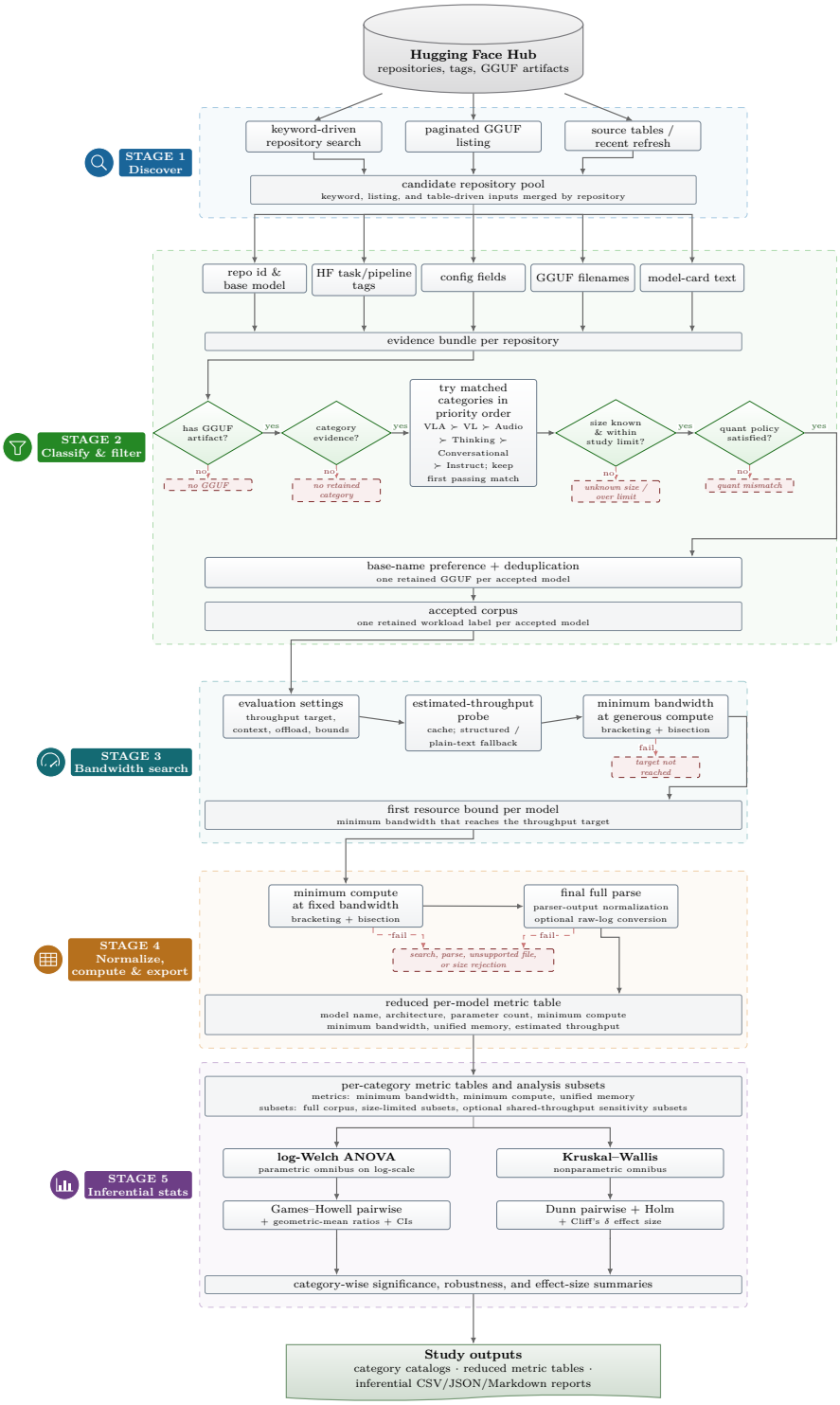

Figure 1. End-to-end methodology workflow used to construct the GGUF corpus, derive category-wise deployment envelopes, and analyze the exported metrics statistically.

Table 2 summarizes the controls that make the manuscript-facing estimates comparable across workload categories. The table links the public HFH GGUF sampling

frame, the Q4-K-M checkpoint policy, the `gguf-parser-go` estimator settings, the throughput regimes, and the statistical analysis layer into one traceable design. These controls operationalize the GGUF-based corpus restriction and parser-based resource estimation described above [10,12].

Table 2. Study-design controls and analysis safeguards used for the manuscript-facing results.

Design Element	Manuscript-Facing Implementation
Sampling frame	Public HFH repositories exposing GGUF artifacts and matching one of the six deployment-facing workload categories.
Inclusion controls	Known model size, parameter count at or below 10 B, Q4-K-M GGUF checkpoint availability, and one primary workload category per retained repository.
Artifact selection	Representative backbone GGUF selected from the Q4-K-M pool; auxiliary GGUF components such as `mmproj` files recorded as packaging evidence but not added to the backbone-side parser estimate.
Parser and runtime assumptions	`gguf-parser-go` v0.24.0, context size 2048, zero graphics processing unit (GPU)-layer offload, and a UMA-style device descriptor expressed as tera floating-point operations per second (TFLOPS) and GBps.
Threshold-search bounds	Bandwidth range 0.001–273 GBps; compute range 0.001–8.1 TFLOPS; relative tolerance 0.01; maximum 100 refinement iterations.
Throughput regimes	Category-specific targets of 5 TPS for conversational, instruct, and audio, 25 TPS for thinking, 30 TPS for VL, and 9 TPS for VLA; independent unified sensitivity run at 10 TPS for all categories.
Result traceability	Category-specific and unified 10-TPS metric exports used the same schema: model identity, architecture, parameter count, minimum GBps, minimum GFLOPS, UMA/MiB, and predicted TPS. The reported tables and figures were generated from these exports and from the corresponding statistical summaries.
Statistical layer	Welch analysis of variance (ANOVA) and Games–Howell contrasts on log-transformed positive metrics, with Kruskal–Wallis and Dunn–Holm tests retained as nonparametric robustness checks.

Candidate repositories were discovered through the HF application programming interface (API) by using a combined retrieval strategy. Search-based discovery used a curated workload vocabulary that mixed category descriptors, task phrases, and known family names for conversational, instruct, thinking, audio, VL, and VLA checkpoints. Each query was expanded to cover variants with and without an explicit GGUF token, and the search pass was complemented by a paginated traversal of the global GGUF listing. This design intentionally favored recall before filtering: broad retrieval was allowed at the discovery stage, whereas category evidence, artifact availability, quantization, parameter limits, and deduplication were enforced later. The implementation hard-coded the discovery vocabulary, task descriptors, and classifier patterns; Appendix B lists the public rule-documentation resources for search terms, pipeline tags, and classification regexes.

Model classification was deterministic and rule based. For every candidate repository, the pipeline assembled evidence from the repository identifier, declared base-model metadata, HF task or pipeline descriptors, tag lists, configuration fields, GGUF artifact names, and structured or free-text model-card metadata. The task descriptors were normalized so that spelling variants based on underscores, hyphens, or spaces mapped to the same evidence token. Audio evidence included speech recognition, text-to-speech, audio generation, speaker, diarization, and sound-event tasks; VL evidence included image-text, image-to-text, video-text, visual-question-answering, and document-question-answering tasks; and VLA evidence included robotics-related descriptors. These structured tags were not always present, so they were treated as high-confidence evidence when available rather than as the only classification source.

The pattern layer then searched the assembled evidence for category-specific cues. The VLA rules used both explicit action-model family cues and a composite rule requiring

visual-language evidence together with action, robotics, control, manipulation, or embodiment evidence. This composite rule was necessary because many VLA repositories advertise themselves through robotics or policy wording rather than through the exact VLA acronym. VL rules emphasized multimodal vision-language family names and image/video/document understanding tasks. Audio rules emphasized speech, sound, music, transcription, synthesis, speaker, and codec terminology. Thinking rules emphasized reasoning, mathematical, proof, verifier, and deliberation terminology. Conversational and instruct rules separated dialogue-oriented chat or assistant evidence from instruction-tuned, supervised fine-tuning , preference-tuned, and task-following evidence. Thus, the classifier combined explicit family names, task labels, and deployment-facing language rather than depending on a single repository field.

When more than one category matched, the corpus used primary-category assignment. The precedence was specificity-first: action and visual categories were resolved before audio, reasoning, conversational, and instruct categories, so that a multimodal or embodied repository was not absorbed by a generic chat or instruction label merely because its model card also mentioned assistant behavior. The alternative multi-label mode was retained in the implementation for extensibility, but it was not used for the manuscript-facing results because the statistical comparisons required disjoint category groups. Every accepted row therefore had one primary workload label, while rejected candidates retained a recorded reason such as missing GGUF artifact, missing size estimate under the default policy, parameter count above the 10 B bound, or quantization mismatch.

Repository filtering removed obvious incompatibilities before parser invocation. Candidate repositories had to expose at least one GGUF checkpoint and had to satisfy the default size bound of 10 B parameters. Unknown-size repositories were dropped by default so that larger models could not enter the bounded corpus through missing metadata. The manuscript-facing category-specific and unified analyses then retained Q4-K-M GGUF checkpoints only. This quantization filter was used as a practical deployment-oriented compromise that typically preserves usable accuracy while reducing footprint substantially relative to full-precision checkpoints within the GGUF ecosystem, and it prevented quantization choice from becoming an additional source of cross-category variation. When several Q4-K-M GGUF artifacts were available for the same repository, auxiliary projection artifacts were separated from candidate backbone artifacts when possible. The representative artifact was then selected from the Q4-K-M pool, prioritizing backbone checkpoints over companion projection GGUFs. Duplicate repository identifiers and near-duplicate base names were collapsed before parser invocation so that each retained family entered profiling once per primary category.

Algorithm 1 summarizes the combined retrieval, classification, artifact selection, filtering, and deduplication logic implemented in the scraper stack. The ordering is important because broad recall is preserved before stricter evidence, size, quantization, and artifact filters are applied. The classifier rules are hard-coded in the implementation for reproducibility; the public resources linked in Appendix B document the active search vocabulary, task or pipeline descriptors, and pattern inventory rather than being read dynamically at run time. The algorithm remains extensible because new categories can be added by editing the in-code discovery phrases, descriptor sets, regular-expression patterns, quantization policy, size policy, specificity order, and representative-artifact selection rule. In primary-category mode, matched categories were evaluated in the order of their specificity, and the first category that also passed the active size and quantization filters was emitted. Thus, a higher-priority category match that failed filtering did not necessarily block emission under a lower-priority matched category. In practical terms, the wall time of this stage scaled with the number of search jobs, listing pages, and repositories requir-

ing metadata enrichment, and was therefore dominated by network I/O rather than by local computation.

Algorithm 1 Corpus Discovery, Filtering, and Representative-GGUF Selection

procedure BUILD_CATEGORY_CORPUS
  $\mathcal{R} \leftarrow \emptyset$
  for each selected workload search phrase q **do**
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ SEARCH_HUB(q, with GGUF filter)
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ SEARCH_HUB(q, without GGUF filter)
  end for
  if paginated or combined discovery is enabled **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ PAGINATE_HUB(GGUF filter)
  end if
  $\mathcal{R} \leftarrow$ MERGE_BY_REPOSITORY($\mathcal{R}$)
  $\mathcal{D} \leftarrow \emptyset$
  for each repository $r \in \mathcal{R}$ **do**
    $E_r \leftarrow$ COLLECT_EVIDENCE(r)
      ▷ repository id, base model, tags, task descriptors, config, card metadata, GGUF names
    $F_r \leftarrow$ LIST_GGUF_ARTIFACTS(r)
    if $F_r = \emptyset$ **then**
      REJECT(r, no GGUF artifact); **continue**
    end if
    $(F_r, F_r^{Q4}, F_r^{Q4KM}) \leftarrow$ CANONICALIZE_GGUF_LISTS(F_r)
    $a_r \leftarrow$ HAS_AUXILIARY_GGUF(F_r)
    $M_r \leftarrow$ MATCH_WORKLOAD_EVIDENCE(E_r)
    if $M_r = \emptyset$ **then**
      REJECT(r, no category evidence); **continue**
    end if
    if primary-category mode is active **then**
      $\mathcal{C}_r \leftarrow$ ORDER_MATCHES_BY_SPECIFICITY(M_r)
    else
      $\mathcal{C}_r \leftarrow M_r$
    end if
    for each category $c \in \mathcal{C}_r$ **do**
      if c is not selected for this run **then**
        continue
      end if
      if PARAMETERS(r) is unknown and unknown-size models are not allowed **then**
        REJECT(r, c, unknown size); **continue**
      end if
      if PARAMETERS(r) $>$ 10 B **then**
        REJECT(r, c, above size bound); **continue**
      end if
      $F_{r,c}^{*} \leftarrow$ APPLY_QUANTIZATION_POLICY($F_r, F_r^{Q4}, F_r^{Q4KM}, c$)
      if $F_{r,c}^{*} = \emptyset$ **then**
        REJECT(r, c, quantization mismatch); **continue**
      end if
      $f_{r,c} \leftarrow$ SELECT_PREFERRED_BACKBONE_GGUF($F_{r,c}^{*}$)
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(r, c, f_{r,c}, a_r, E_r)\}$
      if primary-category mode is active **then**
        break
      end if
    end for
  end for
  $\mathcal{D} \leftarrow$ DEDUPLICATE_BY_CATEGORY_AND_BASE_NAME($\mathcal{D}$)
  $\mathcal{D} \leftarrow$ POPULATE_SELECTED_FILE_AND_URL($\mathcal{D}$)
  return $\mathcal{D}$
end procedure

3.3. Parser-Output Normalization

The textual outputs produced by the parser were normalized through a custom ASCII-table conversion stage. Raw text was segmented into blank-line-delimited blocks, and each block was scanned for rows beginning with the pipe character. Table cells were recovered by stripping boundary delimiters and splitting the remaining content into columns. Each parsed block was flattened into prefixed key-value pairs so that metadata fields and estimate fields remained distinct in the final record structure. For `ESTIMATE` blocks, a fixed header template was imposed because these tables were not consistently self-described across logs. This normalization stage was introduced to ensure that heterogeneous parser outputs could be converted into a single machine-readable schema without loss of provenance.

Parameter strings such as "1.78 B" were converted into raw counts and subsequently mapped into comparable numeric fields. Parsed records were collected across checkpoints, columns were preserved in first-appearance order, and the final tables were sorted first by parameter regime and then by parameter count. The normalized outputs were exported as UTF-8 tables and XLSX workbooks so that statistical summaries and visualizations could be produced consistently across categories.

3.4. Resource-Threshold Estimation

Complexity profiling was performed with `gguf-parser-go` v0.24.0 through a custom optimization wrapper. Each model was supplied through an HF uniform resource locator (URL) resolved from repository metadata. During profiling, checkpoints marked as Q4-K-M quantizations were targeted to standardize quantization across comparisons; this constraint was preserved for the manuscript-facing category-specific and unified 10-TPS analyses. The parser was queried with a device descriptor of the form `<TFLOPS>;<GBps>`, where the two fields represented compute throughput and memory bandwidth. Feasibility was defined as achieving a predicted maximum throughput greater than or equal to the category-specific throughput target under a context size of 2048 and zero GPU-layer offload.

The parser was used here as a deployment-envelope estimator rather than as a substitute for full board-level benchmarking. It operates from GGUF metadata, reading the header and tensor descriptors needed to estimate weights, key–value (KV) cache demand, and activation buffers without downloading the full checkpoint. According to the authors of `gguf-parser-go`, under correctly matched runtime settings such as context size, GPU-layer offload, and cache quantization, the evaluation results usually deviate from actual hardware memory usage by about 100 MiB. The same tool's authors also report validation on both UMA and non-unified memory architecture (NUMA)-style hardware configurations, including Apple Mac Studio (M2), Intel i5-14600K, and NVIDIA GeForce RTX 4080 systems [12]. By contrast, the reported maximum TPS values should be interpreted as best-case theoretical throughput for the supplied device descriptor because realized performance on physical hardware may be lower owing to operating-system overhead, thermal throttling, or backend-specific implementation differences. For that reason, the present work treated parser outputs as screening-level hardware estimates and reserved direct board-level calibration for future validation.

A two-stage minimization strategy was adopted. In the first stage, memory bandwidth was minimized while computational throughput was held at a high fixed value. In the second stage, the minimum compute requirement was minimized while the bandwidth was fixed at the feasible minimum obtained in the first stage. This ordering was deliberate: bandwidth was treated as the more power-sensitive objective for edge systems because reducing required memory traffic generally reduces switching activity and the associated dynamic power in the memory subsystem. Both stages used the same threshold-search routine. A bracketing phase first expanded the test point until the throughput constraint

became feasible or an upper bound was reached. A bisection phase then refined the feasible interval until the relative interval width satisfied the stopping tolerance. This custom search procedure was necessary because the parser estimated throughput for a specified device metric but did not directly return the minimum feasible hardware pair that satisfied the target constraint.

Throughput extraction was made robust to parser-version differences. JavaScript Object Notation (JSON) output was preferred whenever available; however, some parser versions nested the maximum-throughput field under deeper paths such as `maxTps.value` rather than exposing a simple top-level key. For that reason, the JSON tree was traversed by full path, token-throughput candidates were ranked by context, and values associated with latency-related contexts, such as milliseconds (ms), time per token (tpt), or time to first token (ttft), were ignored. When JSON decoding failed, when compact JSON flags were unsupported, or when the estimate contained non-finite throughput values, ASCII parsing was used as a fallback. Parser calls were cached by exact device-metric string to prevent redundant evaluations at previously tested operating points.

After convergence, the parser was executed again at the final operating point in order to recover architecture, parameter count, quantization label, RAM-layer estimates, UMA usage, non-UMA memory usage, and video random access memory (VRAM)-layer estimates. When UMA or non-UMA fields were missing from the full parse, an additional estimate-only pass was issued and the missing values were back-filled. Final export rows contained model name, architecture, parameters in billions, minimum TFLOPS, minimum GBps, UMA memory in MiB, and the predicted throughput at the converged operating point. For tables and plots, TFLOPS values were rescaled to GFLOPS.

Algorithm 2 was invoked twice for each model: first with EVAL(GBps) while TFLOPS was fixed at a high value, and then with EVAL(TFLOPS) while GBps was fixed at the feasible minimum returned by the first invocation. The run-time settings followed the main category-specific configuration, which preserved 5-TPS targets for conversational, instruct, and audio models, 25 TPS for thinking models, 30 TPS for VL models, and 9 TPS for VLA models. These values were treated as scenario parameters rather than universal latency standards: they encoded lower-rate human-facing text and audio interaction, higher-rate hidden deliberation for reasoning-oriented models, elevated visual-token throughput for VL workloads, and a moderate action-generation target for VLA workloads. After bracketing, the refinement stage required logarithmically many parser evaluations in the interval width, while overall wall time remained dominated by parser invocations. The bracketing-plus-bisection structure was retained because it guaranteed a feasible bound whenever the target TPS could be reached, and it exposed the smallest acceptable operating point with a transparent stopping rule.

Post-processing was standardized across all six categories. A separate unified 10-TPS sensitivity export was also analyzed; it used the same metric schema and reporting logic but replaced the category-specific service targets with a common throughput target. Summary tables reported arithmetic means, sample standard deviations, and direct minimum–maximum ranges for the full corpus, the below 4 B parameter subset, and the below 1 B parameter subset. Frontier extraction then restricted each category to below 4 B parameters, min–max normalized GBps and GFLOPS within that filtered subset, summed the two normalized quantities into a combined score, and retained the lowest-scoring model per backbone. Backbone popularity was quantified by architecture-frequency counts within each category and across the pooled corpus.

Algorithm 2 Threshold Minimization for a Single-Resource Variable

procedure SEARCH_THRESHOLD(EVAL, T_{target})
  $lo \leftarrow x_{floor}$
  $hi \leftarrow \min(\max(x_{start}, x_{floor}), x_{cap})$
  $t_{hi} \leftarrow \text{EVAL}(hi)$
  while $t_{hi} < T_{target} \wedge hi < x_{cap}$ **do**
    $lo \leftarrow hi$
    $hi \leftarrow \min(2 \cdot hi, x_{cap})$
    $t_{hi} \leftarrow \text{EVAL}(hi)$
  end while
  if $t_{hi} < T_{target}$ **then**
    return FAILURE
  end if
  if $lo = x_{floor} \wedge lo < hi$ **then**
    $t_{lo} \leftarrow \text{EVAL}(lo)$
    if $t_{lo} \geq T_{target}$ **then**
      return lo
    end if
  end if
  for $k \leftarrow 1$ **to** $N_{\max}$ **do**
    $mid \leftarrow (lo + hi)/2$
    $t_{mid} \leftarrow \text{EVAL}(mid)$
    if $t_{mid} \geq T_{target}$ **then**
      $hi \leftarrow mid$
    else
      $lo \leftarrow mid$
    end if
    if $hi > 0 \wedge (hi - lo)/hi \leq \epsilon$ **then**
      return hi
    end if
  end for
  return hi
end procedure

3.5. Inferential Statistical Analysis

To complement the descriptive tables and frontier rankings, a custom Python inferential module implemented category-wise statistical analysis. The module was designed for the same six metric exports used throughout the paper and accepted category-wise metric tables supplied through explicit mappings. Each input table was required to contain model identity, architecture, parameter count in billions, TFLOPS, GBps, UMA/MiB, and observed TPS fields. The module standardized those columns into an internal schema, converted numeric fields, discarded rows missing the fields required for analysis, derived GFLOPS as 1000 × TFLOPS, and removed non-positive GBps, GFLOPS, or UMA/MiB values so that logarithmic analyses were well defined. Quality-control rows recorded raw row counts, rows retained after numeric conversion, missing-field exclusions, non-positive metric exclusions, and final valid row counts for each category.

Algorithm 3 summarizes the full inferential workflow. The algorithm sits after the parser and threshold-estimation stages: it does not estimate new hardware envelopes, but takes the already exported deployment metrics as input and tests whether category labels are associated with systematic differences in GBps, GFLOPS, and UMA/MiB. This separation kept the resource-estimation stage reproducible and allowed the statistical layer to be re-run under alternative throughput-label regimes, subset definitions, and robustness-check settings without re-inspecting the GGUF checkpoints.

Algorithm 3 Inferential Analysis of Category-Wise Deployment Metrics

procedure RUN_INFERENTIAL_ANALYSIS
  $D \leftarrow \emptyset$
  $Q \leftarrow \emptyset$
  $\mathcal{C} \leftarrow \{\text{Conversational}, \text{Instruct}, \text{Thinking}, \text{Audio}, \text{VL}, \text{VLA}\}$
  $\mathcal{M} \leftarrow \{\text{GBps}, \text{GFLOPS}, \text{UMA/MiB}\}$
  if TPS regime $=$ unified **then**
    for $c \in \mathcal{C}$ **do**
      $T_c \leftarrow T_{\text{unified}}$
    end for
  else
    for $c \in \mathcal{C}$ **do**
      $T_c \leftarrow \text{CATEGORY\_SPECIFIC\_TPS}(c)$
    end for
  end if
  for $c \in \mathcal{C}$ **do**
    $D_c \leftarrow \text{LOAD\_METRIC\_TABLE}(c)$
    $D_c \leftarrow \text{STANDARDIZE\_COLUMNS}(D_c)$
    $D_c \leftarrow \text{CONVERT\_NUMERIC}(D_c)$
    $D_c \leftarrow \text{DROP\_MISSING}(D_c, \{\text{parameter count}, \text{compute}, \text{bandwidth}, \text{UMA memory}\})$
    $D_c[\text{compute in GFLOPS}] \leftarrow 1000 \cdot D_c[\text{compute in TFLOPS}]$
    $D_c \leftarrow \text{KEEP\_POSITIVE}(D_c, \{\text{GBps}, \text{GFLOPS}, \text{UMA/MiB}\})$
    $\text{ATTACH}(D_c, c, T_c)$
    $Q \leftarrow Q \cup \text{QUALITY\_CHECKS}(D_c)$
    $D \leftarrow D \cup D_c$
  end for
  $\mathcal{S} \leftarrow \{\text{ALL}(D), \text{PARAMETERS\_LT}(D, 4\text{B})\}$
  if $P_{\text{small}} > 0$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{PARAMETERS\_LT}(D, P_{\text{small}})\}$
  end if
  for $S \in \mathcal{S}$ **do**
    if $\text{NUMBER\_OF\_PRESENT\_CATEGORIES}(S) < 2$ **then**
      continue
    end if
    for $m \in \mathcal{M}$ **do**
      $\text{SAVE\_DESCRIPTIVE\_STATISTICS}(S, m)$
      $\mathcal{C}^{W}_{S,m} \leftarrow \emptyset$
      for $c \in \mathcal{C}$ **do**
        $V_{c,m} \leftarrow \text{VALUES}(S, c, m)$
        if $|V_{c,m}| \geq 2$ **then**
          $G_c \leftarrow \log(V_{c,m})$
          $\mathcal{C}^{W}_{S,m} \leftarrow \mathcal{C}^{W}_{S,m} \cup \{c\}$
        end if
      end for
      if $|\mathcal{C}^{W}_{S,m}| \geq 2$ **then**
        $W \leftarrow \text{WELCH\_ANOVA}(\{G_c : c \in \mathcal{C}^{W}_{S,m}\})$
        $(\eta^2_{\log}, \omega^2_{\log}) \leftarrow \text{LOG\_SCALE\_EFFECT\_SIZES}(\{G_c : c \in \mathcal{C}^{W}_{S,m}\})$
        $\text{SAVE\_WELCH\_RESULTS}(W, \eta^2_{\log}, \omega^2_{\log}, S, m)$
        for $(c_a, c_b) \in \text{ALL\_PAIRWISE\_CONTRASTS}(\mathcal{C}^{W}_{S,m})$ **do**
          $H_{a,b} \leftarrow \text{GAMES\_HOWELL}(G_{c_a}, G_{c_b})$
          $d_{a,b} \leftarrow \bar{G}_{c_a} - \bar{G}_{c_b}$
          $R_{a,b} \leftarrow \exp(d_{a,b})$
          $\Delta_{a,b} \leftarrow 100 \cdot (R_{a,b} - 1)$
          $[\ell_{a,b}, u_{a,b}] \leftarrow \text{GAMES\_HOWELL\_CI}(G_{c_a}, G_{c_b})$
          $[\exp(\ell_{a,b}), \exp(u_{a,b})] \leftarrow \text{BACKTRANSFORM\_CI}([\ell_{a,b}, u_{a,b}])$
          $\text{SAVE\_GAMES\_HOWELL\_RESULTS}(H_{a,b}, R_{a,b}, \Delta_{a,b}, [\exp(\ell_{a,b}), \exp(u_{a,b})])$
        end for
      end if
      if WITH_NONPARAMETRIC $=$ True **then**
        $\mathcal{C}^{K}_{S,m} \leftarrow \emptyset$
        for $c \in \mathcal{C}$ **do**

Algorithm 3 *Cont.*

$V_{c,m} \leftarrow \text{VALUES}(S, c, m)$
if $|V_{c,m}| \geq 1$ **then**
$\mathcal{C}^K_{S,m} \leftarrow \mathcal{C}^K_{S,m} \cup \{c\}$
end if
end for
if $|\mathcal{C}^K_{S,m}| \geq 2$ **then**
$K \leftarrow \text{KRUSKAL\_WALLIS}(\{V_{c,m} : c \in \mathcal{C}^K_{S,m}\})$
$\epsilon^2 \leftarrow \text{KRUSKAL\_EFFECT\_SIZE}(K)$
$\text{SAVE\_KRUSKAL\_RESULTS}(K, \epsilon^2, S, m)$
$\mathcal{P}_{\text{raw}} \leftarrow \emptyset$
$\mathcal{D}_{\text{pairs}} \leftarrow \emptyset$
for $(c_a, c_b) \in \text{ALL\_PAIRWISE\_CONTRASTS}(\mathcal{C}^K_{S,m})$ **do**
$(z_{a,b}, p^{\text{raw}}_{a,b}) \leftarrow \text{TIE\_ADJUSTED\_DUNN\_PAIRWISE}(S, c_a, c_b, m)$
$\delta_{a,b} \leftarrow \text{CLIFFS\_DELTA}(S, c_a, c_b, m)$
$\Delta^{\text{med}}_{a,b} \leftarrow \text{MEDIAN}(V_{c_a,m}) - \text{MEDIAN}(V_{c_b,m})$
$\mathcal{P}_{\text{raw}} \leftarrow \mathcal{P}_{\text{raw}} \cup \{p^{\text{raw}}_{a,b}\}$
$\mathcal{D}_{\text{pairs}} \leftarrow \mathcal{D}_{\text{pairs}} \cup \{(c_a, c_b, z_{a,b}, p^{\text{raw}}_{a,b}, \delta_{a,b}, \Delta^{\text{med}}_{a,b})\}$
end for
$\mathcal{P}_{\text{Holm}} \leftarrow \text{HOLM\_CORRECTION}(\mathcal{P}_{\text{raw}})$
$\text{SAVE\_DUNN\_HOLM\_RESULTS}(\mathcal{D}_{\text{pairs}}, \mathcal{P}_{\text{Holm}})$
end if
end if
end for
end for
EXPORT(merged metrics, Q, descriptive statistics)
EXPORT(Welch ANOVA, Games–Howell)
if WITH_NONPARAMETRIC = True **then**
EXPORT(Kruskal–Wallis, Dunn–Holm)
end if
EXPORT(metadata, analysis report)
return statistical output tables
end procedure

The inferential module was used in two throughput-label regimes only. In the category-specific regime, the metadata preserved the original target-TPS labels used in the main profiling workflow: 5 TPS for conversational, instruct, and audio models, 25 TPS for thinking models, 30 TPS for VL models, and 9 TPS for VLA models. In the unified regime, a single user-specified target TPS was attached to every category, and the manuscript-facing run used the common 10-TPS target. Thus, the inferential results reported below are restricted to the category-specific analysis and the independent unified 10-TPS sensitivity analysis.

The analyzed inferential subsets were generated automatically. The manuscript-facing subsets were the full retained corpus and the compact subset with below 4 B parameters. For every subset, the module first checked that at least two categories were present. For each of the three metrics, it then exported descriptive statistics comprising sample size, arithmetic mean, sample standard deviation, median, geometric mean, log-scale skewness, log-scale excess kurtosis, first and third quartiles, minimum, and maximum. The log-scale skewness and kurtosis columns were used as scale diagnostics rather than binary normality tests, since formal normality tests become overly sensitive to minor deviations in large corpora.

The primary inferential layer used log-transformed GBps, GFLOPS, and UMA/MiB values because these deployment metrics were strictly positive and empirically right-skewed. The log scale stabilized variance and made pairwise effects interpretable as ratios of geometric means after exponentiation. For each subset–metric combination, categories with at least two retained observations entered a Welch one-way ANOVA, which avoided the equal-variance assumption that would be inappropriate for the strongly unbalanced category sizes in this corpus [17,18]. The module also computed classical

between-group η^2 and ω^2 summaries on the same log scale, but these were retained as descriptive comparability indices rather than interpreted as heteroscedastic variance-explained estimates. Substantive interpretation therefore emphasized geometric-mean ratios, confidence intervals, and the nonparametric effect-size summaries.

When the Welch omnibus test was estimable, the post hoc layer computed all pairwise Games–Howell comparisons among the categories present in that subset–metric combination; the procedure is suitable for unequal sample sizes and unequal variances [19]. The manuscript table reports selected rows from the exported all-pairs table, chosen to compare the heavy visual and reasoning branches against lighter text/audio branches and to test the VLA branch against the conversational baseline. The selected rows are therefore a reporting subset, not a restriction imposed by the statistical script. For each computed pairwise contrast, the module saved the mean log difference, group-specific log-scale standard deviations, standard error, Welch–Satterthwaite degrees of freedom, studentized-range statistic, adjusted probability value, and confidence interval. The log difference $d_{a,b}$ was back-transformed to a geometric-mean ratio $R_{a,b} = \exp(d_{a,b})$, and the percentage change was reported as $100(R_{a,b} - 1)$. Consequently, ratios greater than one indicated that category a required a larger geometric-mean deployment metric than category b for the corresponding subset and metric.

The nonparametric layer was enabled by default for manuscript-facing runs and could be skipped only for faster exploratory runs. It was interpreted as a distribution-free robustness layer that coexisted with, rather than replaced, the primary log-Welch/Games–Howell analysis. It used Kruskal–Wallis omnibus tests on the original metric scale, exported the H statistic and an adjusted ϵ^2 effect-size estimate, and then computed tie-adjusted Dunn rank comparisons for all pairwise category contrasts with Holm correction [20–23]. As with the Games–Howell output, the manuscript reports selected rows from this exported all-pairs robustness table. The nonparametric contrast table also saved median differences and Cliff's delta, with magnitude labels defined for the analysis as negligible for $|\delta| < 0.147$, small for $|\delta| < 0.33$, medium for $|\delta| < 0.474$, and large otherwise [24].

The mathematical quantities used by the inferential module were defined at the subset–metric level. For metric m and category i, let $x_{ij}^{(m)} > 0$ denote the retained value for model j, with $i = 1, \ldots, k$ and $j = 1, \ldots, n_i$. The compute column used in the inferential tables was converted from TFLOPS to GFLOPS, and the primary parametric analysis used the natural logarithm of each positive deployment metric:

$$\mathrm{GFLOPS}_{ij} = 1000\,\mathrm{TFLOPS}_{ij}, \qquad z_{ij}^{(m)} = \log x_{ij}^{(m)}, \qquad \mathrm{GM}_i^{(m)} = \exp\left(\bar{z}_i^{(m)}\right), \tag{1}$$

where $\bar{z}_i^{(m)}$ is the category mean on the log scale and $\mathrm{GM}_i^{(m)}$ is the corresponding geometric mean. For readability, the superscript (m) is omitted in the remaining formulas when the metric is clear from context.

Welch ANOVA was computed from the log-scale group means, variances, and sample sizes. With s_i^2 denoting the sample variance of z_{ij} in category i, the inverse-variance weights and weighted grand mean were

$$w_i = \frac{n_i}{s_i^2}, \qquad W = \sum_{i=1}^{k} w_i, \qquad \bar{z}_{\mathrm{W}} = \frac{\sum_{i=1}^{k} w_i \bar{z}_i}{W}. \tag{2}$$

The Welch statistic and denominator degrees of freedom were then

$$F_W = \frac{\frac{1}{k-1}\sum_{i=1}^{k} w_i(\bar{z}_i - \bar{z}_W)^2}{1 + \frac{2(k-2)}{k^2-1}\sum_{i=1}^{k}\frac{1}{n_i-1}\left(1-\frac{w_i}{W}\right)^2}, \tag{3}$$

$$\nu_1 = k-1, \qquad \nu_2 = \frac{k^2-1}{3\sum_{i=1}^{k}\frac{1}{n_i-1}\left(1-\frac{w_i}{W}\right)^2}, \tag{4}$$

with the omnibus probability value obtained from the F_{ν_1,ν_2} distribution. The log-scale effect-size columns saved by the script used classical between- and within-group sums of squares for descriptive comparability,

$$\begin{aligned} SS_B &= \sum_{i=1}^{k} n_i(\bar{z}_i - \bar{z})^2, \\ SS_W &= \sum_{i=1}^{k}\sum_{j=1}^{n_i}(z_{ij} - \bar{z}_i)^2, \\ SS_T &= SS_B + SS_W, \end{aligned} \tag{5}$$

where $\bar{z}$ is the ordinary grand mean over all retained observations in the subset. These summaries were not treated as Welch-specific variance-explained estimators. The reported comparability indices were

$$\eta^2_{\log} = \frac{SS_B}{SS_T}, \qquad \omega^2_{\log} = \max\left\{0, \frac{SS_B - (k-1)MS_W}{SS_T + MS_W}\right\}, \qquad MS_W = \frac{SS_W}{N-k}. \tag{6}$$

For each Games–Howell comparison between categories a and b, the script used the log-scale mean difference, its standard error, and the Welch–Satterthwaite degrees of freedom:

$$d_{ab} = \bar{z}_a - \bar{z}_b, \qquad SE_{ab} = \sqrt{\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b}}, \tag{7}$$

$$\begin{aligned} \nu_{ab} &= \frac{\left(\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b}\right)^2}{\frac{(s_a^2/n_a)^2}{n_a-1} + \frac{(s_b^2/n_b)^2}{n_b-1}}, \\ q_{ab} &= \frac{|d_{ab}|}{SE_{ab}}\sqrt{2}. \end{aligned} \tag{8}$$

The pairwise probability value was evaluated from the studentized-range distribution with k groups and ν_{ab} degrees of freedom. The confidence interval and interpretive effect were reported after back-transformation from the log scale:

$$\begin{aligned} \mathrm{CI}_{R,ab} = \Bigg[&\exp\left(d_{ab} - q_{1-\alpha;k,\nu_{ab}}\frac{SE_{ab}}{\sqrt{2}}\right), \\ &\exp\left(d_{ab} + q_{1-\alpha;k,\nu_{ab}}\frac{SE_{ab}}{\sqrt{2}}\right)\Bigg], \end{aligned} \tag{9}$$

$$R_{ab} = \exp(d_{ab}), \qquad \Delta_{ab} = 100(R_{ab} - 1). \tag{10}$$

For the nonparametric robustness checks, the Kruskal–Wallis and Dunn procedures operated on pooled ranks of the original metric values. Let $\bar{R}_i$ be the mean rank for category i, $N = \sum_i n_i$, and let t denote the size of a tied rank group, with $\sum_t$ taken over all tied groups. The tie-correction factor and omnibus statistic were

$$C = 1 - \frac{\sum_t (t^3 - t)}{N^3 - N}, \tag{11}$$

$$H = \frac{\frac{12}{N(N+1)} \sum_{i=1}^{k} n_i \left(\bar{R}_i - \frac{N+1}{2} \right)^2}{C}. \tag{12}$$

The nonparametric omnibus effect size was saved as the adjusted epsilon-squared index

$$\epsilon^2_{\text{adj}} = \min\left\{1, \max\left[0, \frac{H - k + 1}{N - k}\right]\right\}. \tag{13}$$

For Dunn pairwise comparisons, the rank statistic for categories a and b was

$$z_{ab} = \frac{\bar{R}_a - \bar{R}_b}{\sqrt{\frac{N(N+1)}{12} C \left(\frac{1}{n_a} + \frac{1}{n_b} \right)}}. \tag{14}$$

If $p_{(r)}$ denotes the rth smallest raw Dunn probability value among all P pairwise Dunn tests for a subset–metric combination, the Holm-adjusted sequence was

$$p^{\text{Holm}}_{(r)} = \max_{q \le r} \min\left\{1, (P - q + 1) p_{(q)}\right\}. \tag{15}$$

Finally, pairwise stochastic dominance was summarized as Cliff's delta using the Mann–Whitney statistic U_{ab}:

$$\delta_{ab} = \frac{2U_{ab}}{n_a n_b} - 1. \tag{16}$$

The output contract was deliberately explicit. Each run wrote the merged metric table, per-category quality checks, descriptive statistics with log-scale distribution diagnostics, Welch ANOVA results, Games–Howell contrast results, and, by default, Kruskal–Wallis and Dunn–Holm robustness tables. It also wrote a JSON metadata record containing input-category mappings, category counts, parameter thresholds, TPS regime, design target TPS by category, the significance level α, whether nonparametric checks were enabled, and active Python 3.10.11, NumPy 2.2.6, pandas 2.2.3, SciPy 1.15.2, statsmodels 0.14.6, and openpyxl 3.1.5 versions. A Markdown report summarized the run configuration, quality checks, subset sizes, omnibus tests, reported manuscript-facing post hoc rows, the interpretation of classical effect-size columns, and cautions that large samples make statistical significance easier to obtain than substantive importance. Because the number of categories was fixed at six, the local statistical cost scaled approximately linearly with the number of retained rows, analyzed subsets, and all-pairs contrast count.

4. Experimentation and Results

After applying the category-specific throughput targets and parser convergence outputs, the six metric exports contained 21,039 profiled entries in total, including 7551 models with below 4 B parameters and 1806 models with below 1 B parameters. Across the pooled corpus, 101 distinct backbone labels were observed. The category-specific regime used separate service targets: 5 TPS for conversational, instruct, and audio models, 25 TPS for thinking models, 30 TPS for VL models, and 9 TPS for VLA models. These values are therefore interpreted as deployment scenarios rather than as a claim that all categories should be operated at the same user-facing rate. A separate unified sensitivity run, discussed below, used a common 10-TPS target for all categories.

The main results can be summarized before the detailed category views. Under category-specific targets, VL and thinking models occupied the highest mean compute–

bandwidth region, VLA formed a smaller elevated multimodal-action branch, and audio produced the lightest mean envelope. In the below 4 B subset, the same qualitative split remained, showing that the effect was not caused only by near-10 B checkpoints. The unified 10-TPS sensitivity run compressed the compute–bandwidth separation substantially, indicating that service-rate assumptions were a major contributor to main-regime asymmetry. UMA memory varied less strongly than compute or bandwidth, and the VLA category remained the least stable statistically because only 48 retained entries were available.

The results are organized to keep every graphical and tabular output interpretable in context. Each workload category is first described through its descriptive statistics, two diagnostic scatter plots, and a compact backbone-unique frontier. The cross-category figures then aggregate the same metrics to show which asymmetries persist after pooling. The final result subsections report the unified 10-TPS sensitivity analysis, the selected manuscript-facing inferential rows, and an auxiliary context-window analysis for VL checkpoints. This ordering separates observed deployment envelopes from interpretation while ensuring that the visual results, descriptive summaries, and statistical tests are read together rather than as disconnected artifacts.

4.1. Conversational Workload Results

The conversational sample dominated the retained corpus numerically: Table 3 reports 15,205 profiled entries, including 5184 below 4 B parameters. Pruning to the compact subset reduced the average throughput-side envelope from 22.46 GBps and 252.53 GFLOPS to 10.09 GBps and 97.57 GFLOPS. UMA memory moved less, from 815.60 to 747.89 MiB, which indicates that a smaller parameter count reduced compute and bandwidth more strongly than residency.

Table 3. Conversational deployment metrics under the 5-TPS category-specific target. Parameter minima below 0.01 B are displayed as <0.01 rather than rounded to zero.

Regime	Models	Param./B ($\mu \pm \sigma$ [Min, Max])	GFLOPS ($\mu \pm \sigma$ [Min, Max])	GBps ($\mu \pm \sigma$ [Min, Max])	UMA/MiB ($\mu \pm \sigma$ [Min, Max])
Overall	15,205	5.55 ± 2.88 [<0.01, 10.00]	252.53 ± 133.67 [0.03, 656.54]	22.46 ± 10.86 [0.03, 75.71]	815.60 ± 325.53 [15.40, 2088.96]
<4 B parameters	5184	1.91 ± 1.18 [<0.01, 3.90]	97.57 ± 76.54 [0.03, 373.75]	10.09 ± 6.58 [0.03, 32.26]	747.89 ± 297.59 [15.40, 1669.12]
<1 B parameters	1293	0.46 ± 0.21 [<0.01, 0.90]	26.04 ± 18.83 [0.03, 105.80]	3.70 ± 2.06 [0.03, 10.20]	602.99 ± 287.05 [15.40, 1249.28]

The conversational compute–bandwidth panel in Figure 2 has a diagonal structure, meaning that estimated arithmetic demand and memory traffic increased together rather than independently. A large low-resource mass sits below the plotted reference lines, but the upper tail extends to approximately 75.71 GBps and 656.54 GFLOPS. The conversational label therefore spans tiny demonstration checkpoints as well as near-10 B instruction/chat models.

The memory view in Figure 3 explains why parameter count alone is an unreliable feasibility proxy. Although the below 4 B subset reduced mean bandwidth and compute by more than half, the corresponding mean UMA requirement fell only from 815.60 to 747.89 MiB. Similar parameter counts can therefore be mapped to very different residency estimates because the parser budget includes model structure and runtime buffers in addition to raw parameter storage.

The compact conversational frontier in Table 4 is deliberately backbone-unique rather than frequency-weighted. Llama accounted for 49.56% of retained conversational rows across 83 backbone labels, but the frontier itself contained one llama entry alongside granite,

granitemoe, bert, and mamba entries [25–28]. Thus, the minimum feasible conversational cases came from several small model families, while the dominant backbone share described corpus composition rather than the absolute frontier floor.

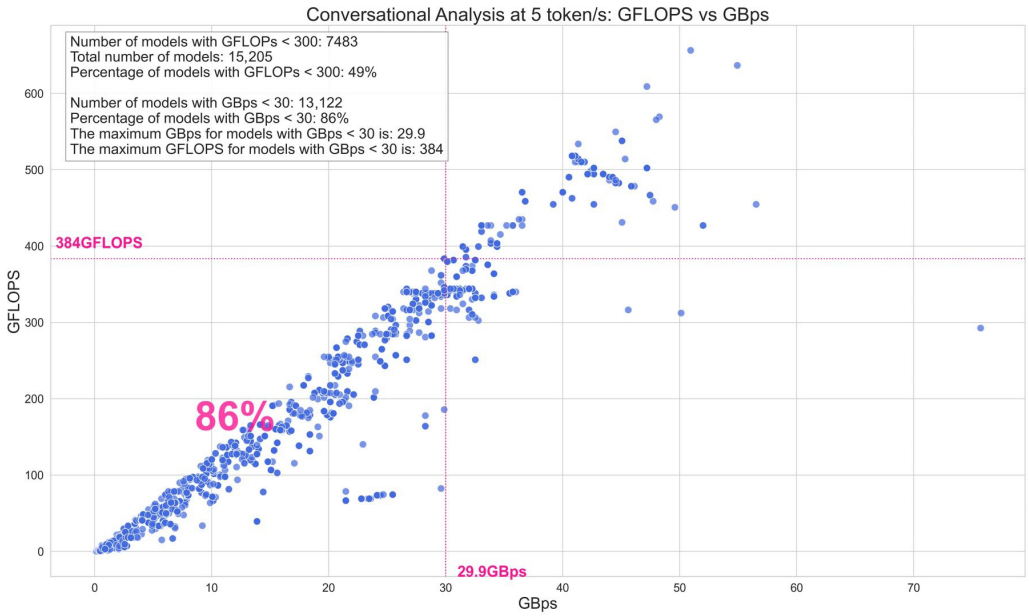

Figure 2. Conversational 5-TPS compute–bandwidth envelope, spanning tiny demonstration checkpoints and near-10 B chat models.

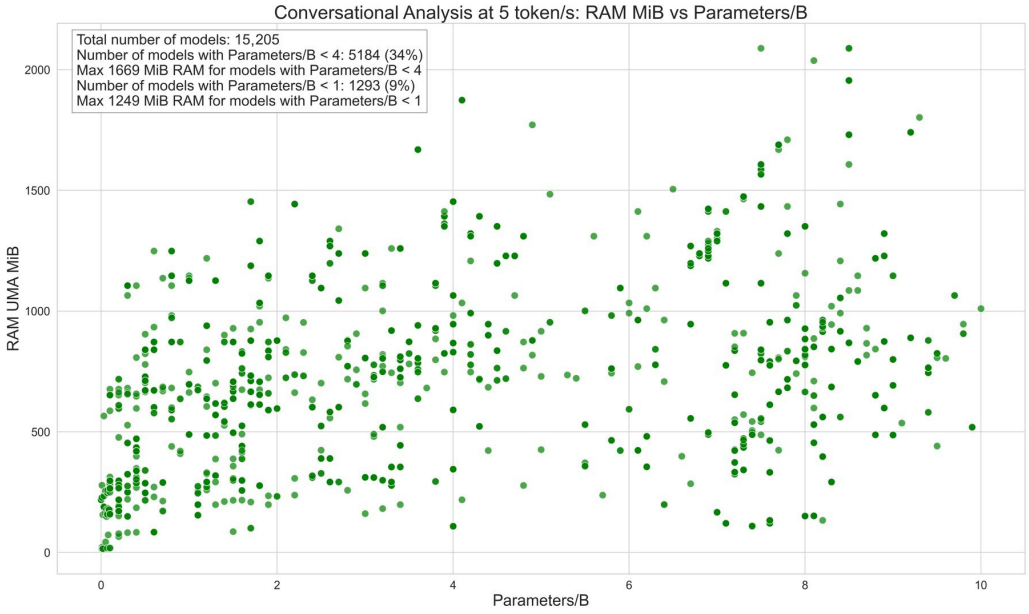

Figure 3. Conversational parameter–UMA scatter, where compact checkpoints can still require substantial residency.

Table 4. Backbone-unique conversational frontier for models with below 4 B parameters in the category-specific run. Parameter counts below 0.01 B are displayed as <0.01.

Model	Backbone	Param./B	GBps	GFLOPS	UMA/MiB
tiny-random-granite-i1-GGUF	granite	<0.01	0.029	0.033	217.44
tiny-random-granite-moe-GGUF	granitemoe	<0.01	0.058	0.063	218.38
NeuroBERT-Mini-GGUF	bert	0.01	0.069	0.096	15.40
tiiuae_-_falcon-mamba-tiny-dev-gguf	mamba	0.01	0.077	0.140	278.84
Cambridge-KAIST2_-_SmolLM-14m-Dolma-v0.1-Base-tied-gguf	llama	0.01	0.176	0.303	227.90

4.2. Instruct Workload Results

The instruct branch was much smaller than the conversational branch, with 581 retained entries in Table 5. Its overall average operating point was 25.61 GBps and 304.39 GFLOPS, whereas the below 4 B subset fell to 8.53 GBps and 81.54 GFLOPS. The accompanying 635.35 MiB compact mean UMA prevents the category from being treated as uniformly memory-light.

Table 5. Instruct deployment metrics under the 5-TPS category-specific target.

Regime	Models	Param./B ($\mu \pm \sigma$ [Min, Max])	GFLOPS ($\mu \pm \sigma$ [Min, Max])	GBps ($\mu \pm \sigma$ [Min, Max])	UMA/MiB ($\mu \pm \sigma$ [Min, Max])
Overall	581	5.93 ± 2.53 [0.02, 9.20]	304.39 ± 146.16 [0.06, 656.54]	25.61 ± 11.60 [0.06, 76.78]	725.55 ± 397.59 [15.94, 2088.96]
<4 B parameters	133	1.59 ± 1.07 [0.02, 3.90]	81.54 ± 64.92 [0.06, 257.08]	8.53 ± 5.85 [0.06, 25.73]	635.35 ± 377.16 [19.96, 1454.08]
<1 B parameters	37	0.39 ± 0.26 [0.02, 0.80]	19.91 ± 15.21 [0.06, 52.40]	3.03 ± 2.25 [0.06, 7.06]	466.81 ± 429.12 [19.96, 1249.28]

In Figure 4, instruct checkpoints occupy a tighter operating cloud than the reasoning and visual categories, mostly inside the low-to-middle part of the threshold plane. The diagonal trend persists, but the spread is narrower than in the conversational branch because the retained instruct sample is smaller and more concentrated in llama-derived families [28]. The largest points still reach 76.78 GBps and 656.54 GFLOPS, so instruction tuning alone is not enough to make a checkpoint edge-light.

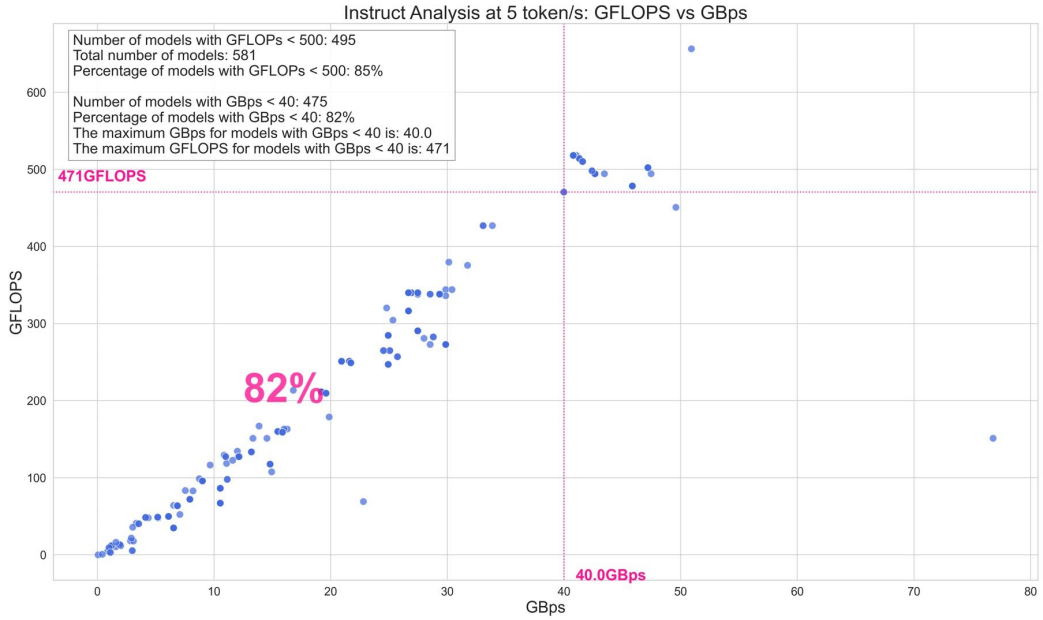

Figure 4. Instruct 5-TPS compute–bandwidth envelope, concentrated in a lower text-centered operating band.

In Figure 5, the memory spread remains visible even at small parameter counts. The below 1 B subset averaged only 19.91 GFLOPS and 3.03 GBps, yet its UMA interval extended from 19.96 to 1249.28 MiB. For instruction-following checkpoints, the practical lesson is therefore not merely to choose a small model, but also to verify the estimated memory envelope separately.

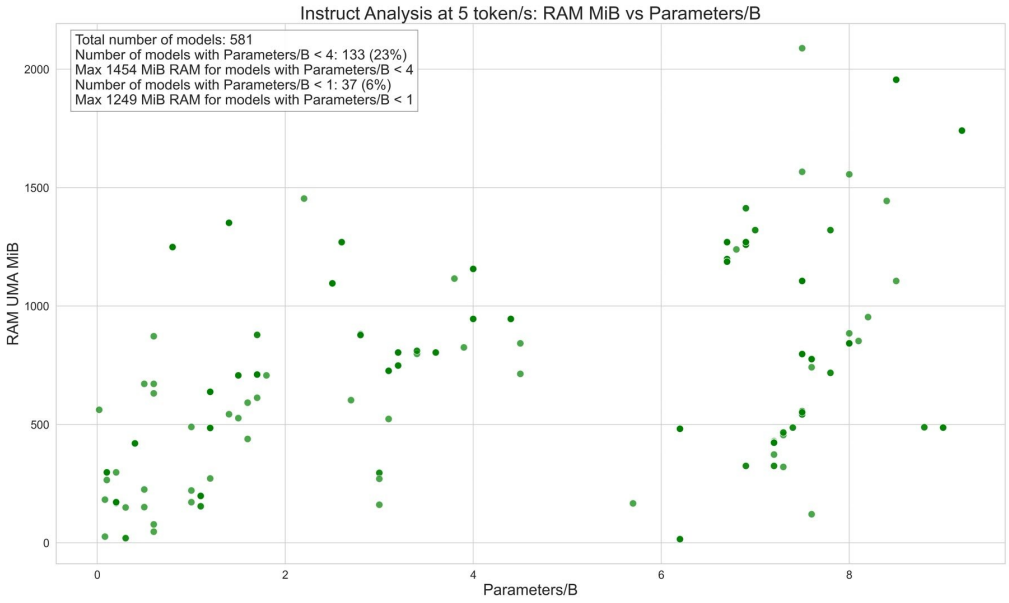

Figure 5. Instruct parameter–UMA scatter, used to verify memory variation within small instruction-tuned checkpoints.

Table 6 highlights how far the best compact instruct cases sit below the category mean. The leading entry required only 0.056 GBps and 0.065 GFLOPS, but the remaining backbone representatives quickly moved into the 0.4–2.0 GBps and 0.7–13.3 GFLOPS range. Because llama supplied 76.59% of the full instruct rows across 26 unique backbone labels, the category average was pulled by larger llama-family checkpoints even though the absolute frontier floor was very low [28].

Table 6. Backbone-unique instruct frontier for models with below 4 B parameters in the category-specific run.

Model	Backbone	Param./B	GBps	GFLOPS	UMA/MiB
Nagi-ovo_-_Nagi_TinyLLaMA_medical_sft-gguf	llama	0.02	0.056	0.065	562.31
rejection_detection-GGUF	bert	0.08	0.412	0.730	26.28
OpenELM-270M-Instruct-GGUF	openelm	0.30	1.000	9.270	149.77
gpt2-rlhf-anthropic-GGUF	gpt2	0.10	2.000	11.989	297.71
mnoukhov_-_pythia160m-sft-tldr-gguf	gptneox	0.20	1.916	13.348	297.90

4.3. Thinking Workload Results

Thinking workloads occupied one of the heaviest regions in the category-specific regime. As reported in Table 7, the category averaged 104.05 GBps and 1153.98 GFLOPS overall, while the compact subset still averaged 49.88 GBps and 482.42 GFLOPS. This persistent separation was consistent with the higher reasoning-oriented throughput target and with the broad upper tail visible in Figure 6.

Table 7. Thinking deployment metrics under the 25-TPS reasoning-oriented target.

Regime	Models	Param./B ($\mu \pm \sigma$ [Min, Max])	GFLOPS ($\mu \pm \sigma$ [Min, Max])	GBps ($\mu \pm \sigma$ [Min, Max])	UMA/MiB ($\mu \pm \sigma$ [Min, Max])
Overall	3324	5.16 ± 2.84 [0.02, 9.80]	1153.98 ± 627.14 [1.01, 3021.68]	104.05 ± 50.82 [0.76, 253.81]	829.36 ± 273.00 [16.18, 2037.76]
<4 B parameters	1270	1.99 ± 1.09 [0.02, 3.90]	482.42 ± 324.72 [1.01, 1692.77]	49.88 ± 28.66 [0.76, 173.82]	740.66 ± 257.53 [16.18, 1669.12]
<1 B parameters	221	0.50 ± 0.20 [0.02, 0.90]	151.21 ± 102.24 [1.01, 482.52]	20.86 ± 11.69 [0.76, 54.92]	628.41 ± 290.85 [16.18, 1167.36]

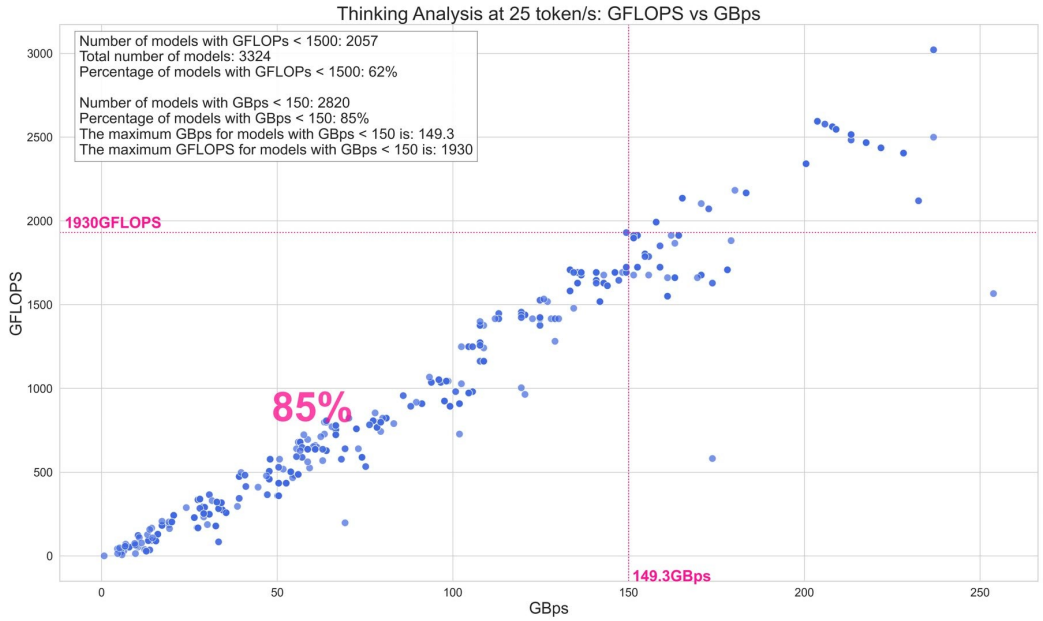

Figure 6. Thinking 25-TPS compute–bandwidth envelope, emphasizing the elevated reasoning-oriented operating region.

The broad elevated cloud in Figure 6 matches the large standard deviations in Table 7. Its main band sits above the conversational and instruct bands because the thinking branch used a higher target throughput and contained many reasoning- or math-oriented language checkpoints with substantial backbone cost. The upper tail reaches 253.81 GBps and 3021.68 GFLOPS, making thinking one of the clearest contributors to the high-resource half of the corpus.

The memory scatter for thinking models in Figure 7 is less categorical than the compute–bandwidth scatter. Below 4 B parameters, the mean UMA was 740.66 MiB and individual estimates ranged from 16.18 to 1669.12 MiB. This separates the throughput-driven elevation of the reasoning branch from a memory profile that still depends heavily on architecture-specific buffers and metadata-derived model structure.

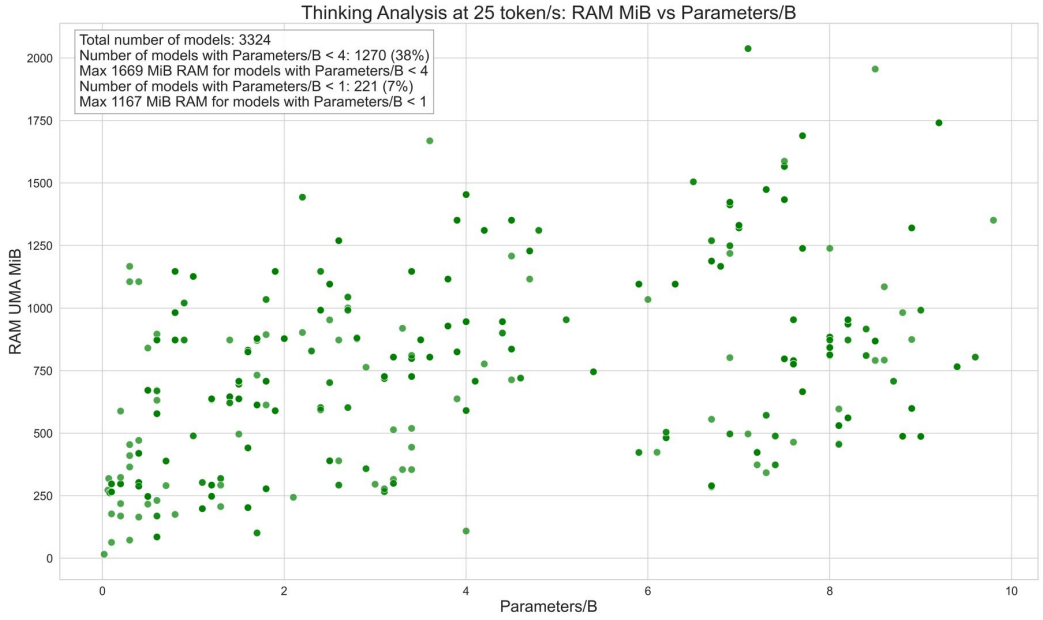

Figure 7. Thinking parameter–UMA scatter, separating the throughput-driven increase from architecture-specific memory variation.

The thinking frontier in Table 8 begins above the text-oriented frontier floors. Its lightest listed point already required 0.762 GBps at the category-specific target, and the next backbone representatives increased quickly in bandwidth, compute, or both. With 44 unique backbone labels and llama contributing 36.49% of rows [28], the compact frontier indicates that the elevated reasoning envelope was not solely an artifact of the largest checkpoints.

Table 8. Backbone-unique thinking frontier for models with below 4 B parameters in the category-specific run.

Model	Backbone	Param./B	GBps	GFLOPS	UMA/MiB
primary-school-math-question-i1-GGUF	bert	0.02	0.762	1.012	16.18
LLaMA3-Reasoning-GGUF	llama	0.20	4.566	14.214	588.48
mmBERT-small-GGUF	modern-bert	0.10	5.832	6.860	63.82
humanoid-logic-model-GGUF	gpt2	0.08	5.465	23.607	261.57
Falcon-H1-Tiny-R-90M-GGUF	falcon-h1	0.10	4.532	42.764	177.66

4.4. Audio Workload Results

Audio occupied the lowest average compute–bandwidth region in Table 9. The full branch averaged 13.41 GBps and 134.03 GFLOPS, and the compact subset stayed comparatively close at 9.45 GBps and 84.32 GFLOPS. Its memory behavior was different: 826.00 MiB overall and 781.42 MiB below 4 B, so low arithmetic and bandwidth demand did not translate into a uniformly small UMA footprint.

Table 9. Audio deployment metrics under the 5-TPS category-specific target.

Regime	Models	Param./B ($\mu \pm \sigma$ [Min, Max])	GFLOPS ($\mu \pm \sigma$ [Min, Max])	GBps ($\mu \pm \sigma$ [Min, Max])	UMA/MiB ($\mu \pm \sigma$ [Min, Max])
Overall	451	2.86 ± 2.44 [0.07, 9.20]	134.03 ± 119.35 [2.60, 518.12]	13.41 ± 10.08 [0.29, 45.86]	826.00 ± 306.89 [14.66, 1740.80]
<4 B parameters	351	1.77 ± 1.33 [0.07, 3.90]	84.32 ± 66.28 [2.60, 330.25]	9.45 ± 6.37 [0.29, 27.46]	781.42 ± 253.48 [14.66, 1269.76]
<1 B parameters	156	0.48 ± 0.16 [0.07, 0.90]	22.46 ± 16.79 [2.60, 97.89]	3.37 ± 1.91 [0.29, 7.60]	641.93 ± 278.04 [14.66, 1157.12]

Figure 8 places most audio checkpoints below the visual and thinking regimes on the compute–bandwidth plane. Audio had the smallest overall mean bandwidth and compute values, but the points still span from very small codec-like checkpoints to larger audio-language models with hundreds of GFLOPS of estimated compute demand. The plotted distribution therefore supports treating the audio branch as light on average, not uniformly light.

Figure 9 makes the throughput/memory mismatch explicit for the audio parameter–UMA view. Audio averaged 826.00 MiB overall, essentially matching the thinking and conversational memory scale despite much lower mean compute. The compact audio subset also retained a mean UMA of 781.42 MiB. Audio checkpoints may therefore fit a modest compute–bandwidth budget while still challenging memory-constrained boards, especially when codec, acoustic, or speech-related components increase residency relative to parameter count.

The audio frontier in Table 10 contained `wavtokenizer_large`, llama, falcon-h1, qwen3, and bert representatives, mixing codec-like, speech/text, and language-backbone cases rather than a single architectural pattern [26,28–31]. The lowest point required 0.285 GBps and 3.399 GFLOPS, yet the five frontier entries ranged from 14.66 to 926.38 MiB in UMA. That spread explains why the audio branch can be light on compute while re-

maining diverse in memory cost; the full category contained 27 backbone labels, with llama representing 44.79% of rows.

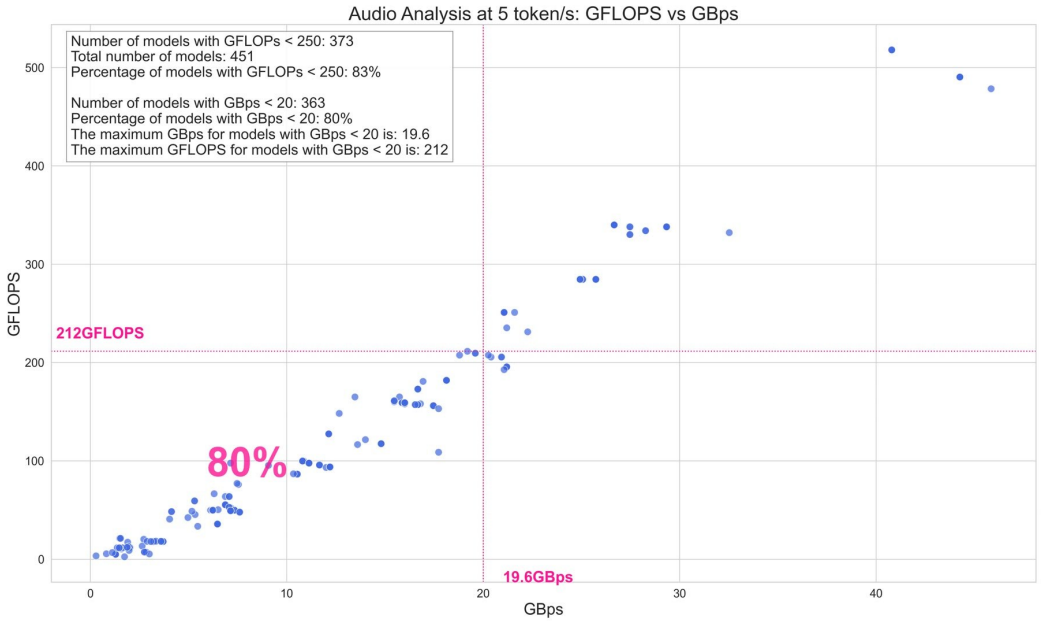

Figure 8. Audio 5-TPS compute–bandwidth envelope, with a low average profile and a heterogeneous upper tail.

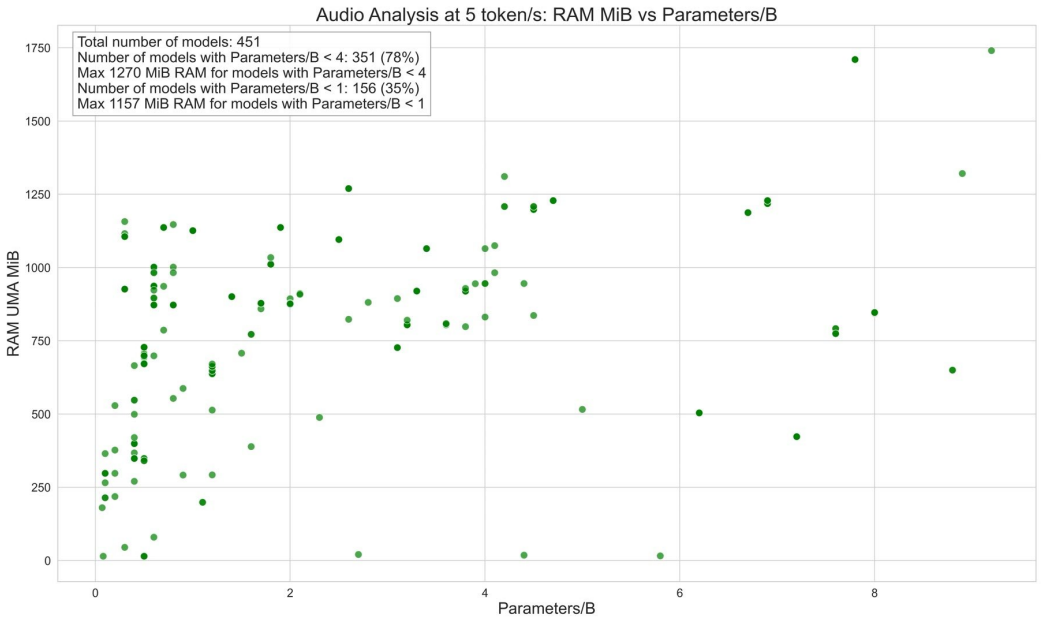

Figure 9. Audio parameter–UMA scatter, highlighting memory spread despite modest compute and bandwidth demand.

Table 10. Backbone-unique audio frontier for models with below 4 B parameters in the category-specific run.

Model	Backbone	Param./B	GBps	GFLOPS	UMA/MiB
codec.cpp-gguf	wavtokenizer_large	0.08	0.285	3.399	14.66
DAC60M-Q4_K_M-GGUF	llama	0.07	0.808	5.531	180.27
MioTTS-GGUF	falcon-h1	0.10	1.116	6.860	364.81
TTS-Encodec-Meta-GGUF	qwen3	0.30	1.283	5.129	926.38
hate-speech-v2-xlm-GGUF	bert	0.30	1.733	2.596	44.77

4.5. Vision-Language Results

Vision-language workloads produced the largest mean envelope in Table 11. Overall requirements reached 119.26 GBps and 1287.85 GFLOPS, and the below 4 B subset still averaged 62.20 GBps and 586.59 GFLOPS. VL also had the highest mean UMA value, 869.46 MiB, indicating that the visual branch affected both the rate-limited operating point and the residency estimate.

Table 11. Vision-language deployment metrics under the 30-TPS visual-token target.

Regime	Models	Param./B ($\mu \pm \sigma$ [Min, Max])	GFLOPS ($\mu \pm \sigma$ [Min, Max])	GBps ($\mu \pm \sigma$ [Min, Max])	UMA/MiB ($\mu \pm \sigma$ [Min, Max])
Overall	1430	5.08 ± 2.80 [0.10, 9.40]	1287.85 ± 683.95 [55.37, 3100.78]	119.26 ± 56.30 [8.06, 245.27]	869.46 ± 253.12 [16.06, 1433.60]
<4 B parameters	593	2.16 ± 1.13 [0.10, 3.90]	586.59 ± 319.42 [55.37, 2025.00]	62.20 ± 31.47 [8.06, 162.09]	813.62 ± 281.05 [16.06, 1413.12]
<1 B parameters	99	0.53 ± 0.23 [0.10, 0.90]	166.24 ± 90.31 [55.37, 340.14]	22.84 ± 11.26 [8.06, 42.39]	624.56 ± 328.09 [16.06, 1146.88]

The VL compute–bandwidth cloud in Figure 10 is both elevated and widely dispersed. Unlike the text/audio categories, the VL distribution begins at a comparatively high compute minimum: even the smallest retained points required 55.37 GFLOPS, and the full category extended to 3100.78 GFLOPS. With a mean bandwidth of 119.26 GBps, the visual branch became the heaviest average operating region under the category-specific scenario.

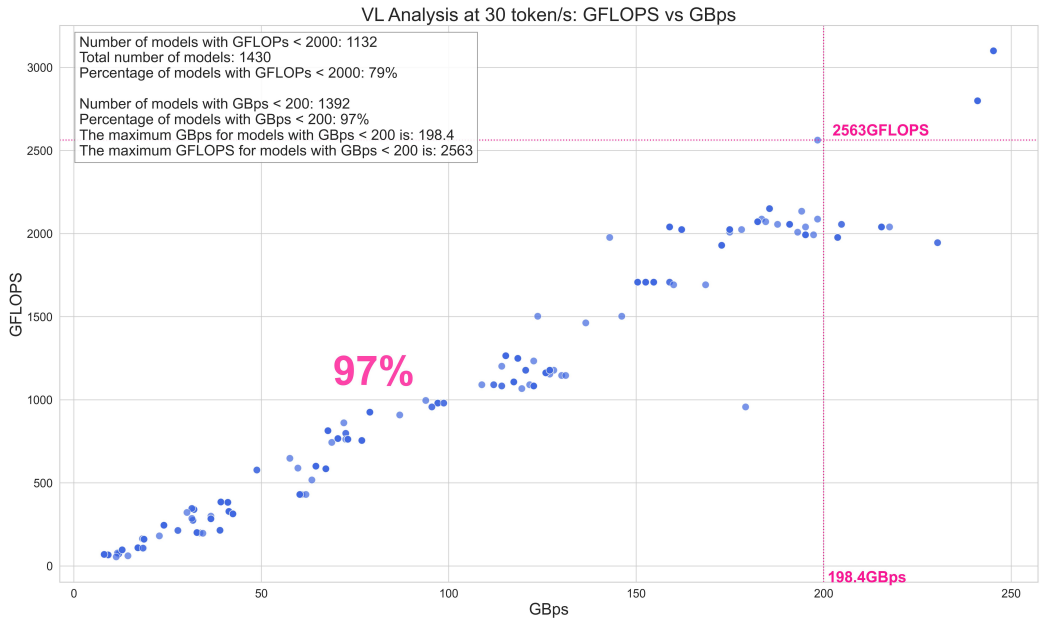

Figure 10. Vision-language 30-TPS compute–bandwidth envelope across the elevated visual operating region.

High memory demand persisted even among smaller visual checkpoints in Figure 11. The below 4 B subset retained a mean UMA of 813.62 MiB, and the below 1 B subset still averaged 624.56 MiB. Reducing the language-backbone parameter count therefore did not eliminate memory pressure in the visual branch; visual conditioning, projection structure, and backbone family still shaped the envelope even when the estimate was anchored to the parsed GGUF backbone.

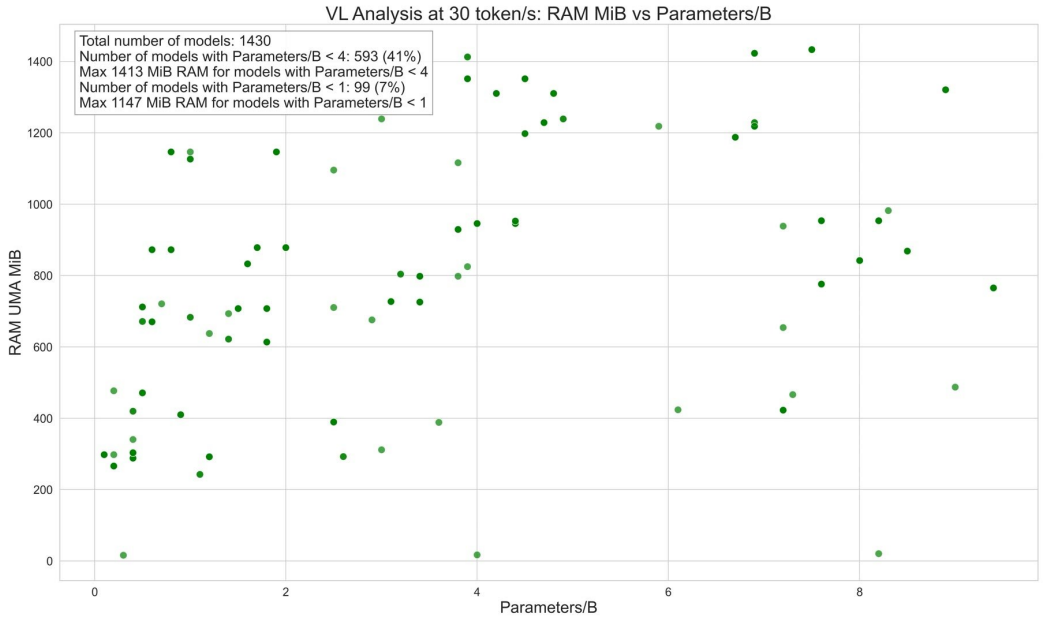

Figure 11. Vision-language parameter–UMA scatter, indicating persistent memory pressure among compact visual checkpoints.

The compact VL frontier in Table 12 remains far above the text/audio floors. Its lightest listed point required 8.065 GBps and 69.708 GFLOPS, even after the below 4 B restriction and backbone-unique selection. The full VL category contained 27 backbone labels and was led by qwen2vl at 39.79% of rows [13]; the frontier confirms that the visual branch stayed separated even at the low-resource edge of the retained sample.

Table 12. Backbone-unique vision-language frontier for models with below 4 B parameters in the category-specific run.

Model	Backbone	Param./B	GBps	GFLOPS	UMA/MiB
AKT-caption-v0.1-lfm2-350m-gguf	lfm2	0.40	8.065	69.708	288.24
SmolDocling-256M-preview-GGUF	llama	0.20	8.931	69.214	266.14
OCRonos-Vintage-GGUF	gpt2	0.10	11.930	72.675	297.71
docling-gguf	pig	0.30	14.396	61.798	16.06
PaddleOCR-VL-0.9B-GGUF	paddleocr	0.50	12.864	97.394	471.47

4.6. Auxiliary Vision-Language Context-Window Results

The VL branch also included an auxiliary context-window comparison. This analysis was not used to define the main category-specific cross-category statistics because the long-context subset was much smaller and represented a different operating scenario. It is nevertheless useful as a stress test of the memory and compute interpretation: a larger context window can change resource demand even when the category label and parser workflow remain fixed.

The long-context VL subset occupied a different part of the compute–bandwidth plane than the larger 2k-context VL corpus. Figure 12 compares 44 models at 128k context with 1430 models at 2k context. The long-context subset had a higher median bandwidth requirement, 161.0 GBps versus 125.8 GBps, and a higher 90th-percentile bandwidth value, 196.2 GBps versus 182.4 GBps. At the same time, its median and 90th-percentile compute values were lower, 1048 and 1210 GFLOPS, compared with 1250 and 2072 GFLOPS for the 2k-context corpus. The mixed direction reflects selection effects as well as context costs because the 128k subset contained different and often smaller checkpoints.

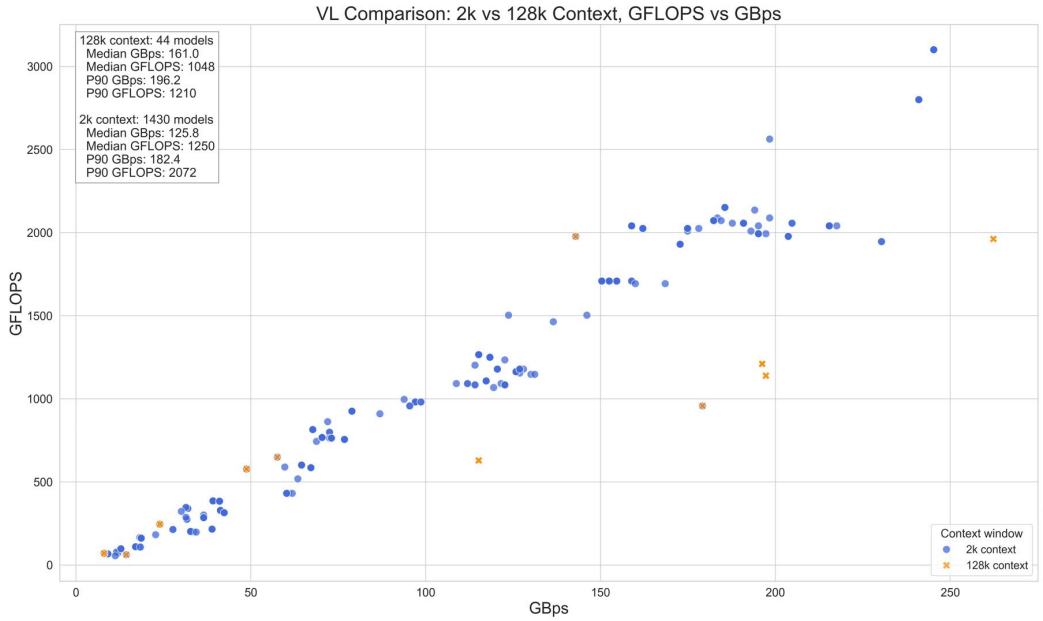

Figure 12. Vision-language context-window comparison on the compute–bandwidth plane. The figure contrasts the standard 2k-token context setting with a 128k-token context setting for the retained VL subset.

The parameter–UMA comparison gives the clearest consequence of extending the context window. In Figure 13, memory demand increases sharply despite the long-context subset having a much smaller median parameter count. The 128k-context subset had a median parameter count of 0.60 B and a median UMA requirement of 1997 MiB, whereas the 2k-context corpus had a median parameter count of 4.40 B and a median UMA requirement of 776 MiB. The 90th-percentile UMA requirement also rose from 1311 to 2396 MiB, strengthening the conclusion that context length, cache-related residency, and model-family composition can dominate raw parameter count.

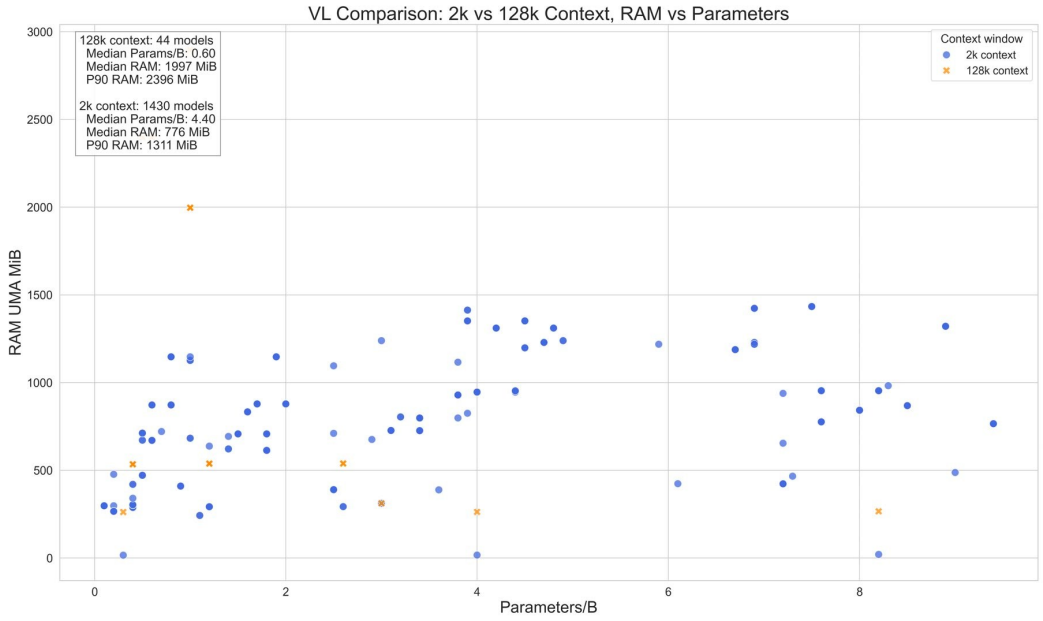

Figure 13. Vision-language context-window comparison for parameter count and UMA. The figure compares the same standard and long-context regimes as Figure 12.

4.7. Vision-Language-Action Results

Vision-language-action (VLA) workloads formed a small but elevated multimodal branch. Table 13 contains 48 retained entries, including 20 below 4 B parameters and none below 1 B. Overall means reached 36.36 GBps and 401.48 GFLOPS, while the compact subset still required 21.24 GBps and 216.54 GFLOPS on average.

Table 13. Vision-language-action deployment metrics under the 9-TPS action-oriented target and sparse retained sample.

Regime	Models	Param./B ($\mu \pm \sigma$ [Min, Max])	GFLOPS ($\mu \pm \sigma$ [Min, Max])	GBps ($\mu \pm \sigma$ [Min, Max])	UMA/MiB ($\mu \pm \sigma$ [Min, Max])
Overall	48	5.26 ± 2.44 [1.70, 8.20]	401.48 ± 182.60 [175.01, 933.40]	36.36 ± 14.58 [19.33, 73.58]	834.77 ± 108.56 [725.91, 1187.84]
<4 B parameters	20	2.83 ± 0.57 [1.70, 3.40]	216.54 ± 22.88 [175.01, 229.39]	21.24 ± 0.99 [19.33, 21.86]	765.16 ± 68.12 [725.91, 886.73]
<1 B parameters	0	–	–	–	–

The VLA operating cloud in Figure 14 is sparse but elevated. Because the branch had only 48 retained entries, the plotted pattern is less stable than the dense conversational, thinking, or VL distributions. Nevertheless, all retained VLA points sat above 19.33 GBps and 175.01 GFLOPS, and the mean values of 36.36 GBps and 401.48 GFLOPS placed the category above the conversational, instruct, and audio means in the category-specific regime.

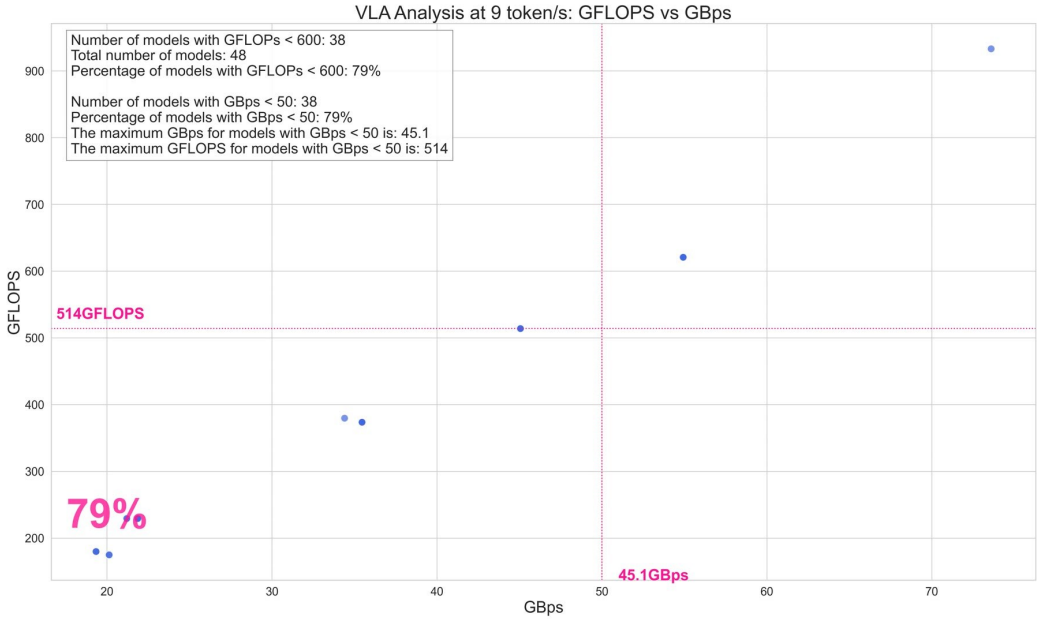

Figure 14. Vision-language-action 9-TPS compute–bandwidth envelope for the sparse retained action-oriented sample.

Figure 15 indicates that memory sensitivity persisted in the retained action-oriented multimodal checkpoints. No VLA model fell below 1 B parameters, and the below 4 B subset had a narrow but non-trivial mean UMA of 765.16 MiB. Because the category was concentrated in a few visual backbones, the VLA scatter should be read as evidence for an elevated but undersampled multimodal-action branch rather than as a stable population estimate for all action-capable models.

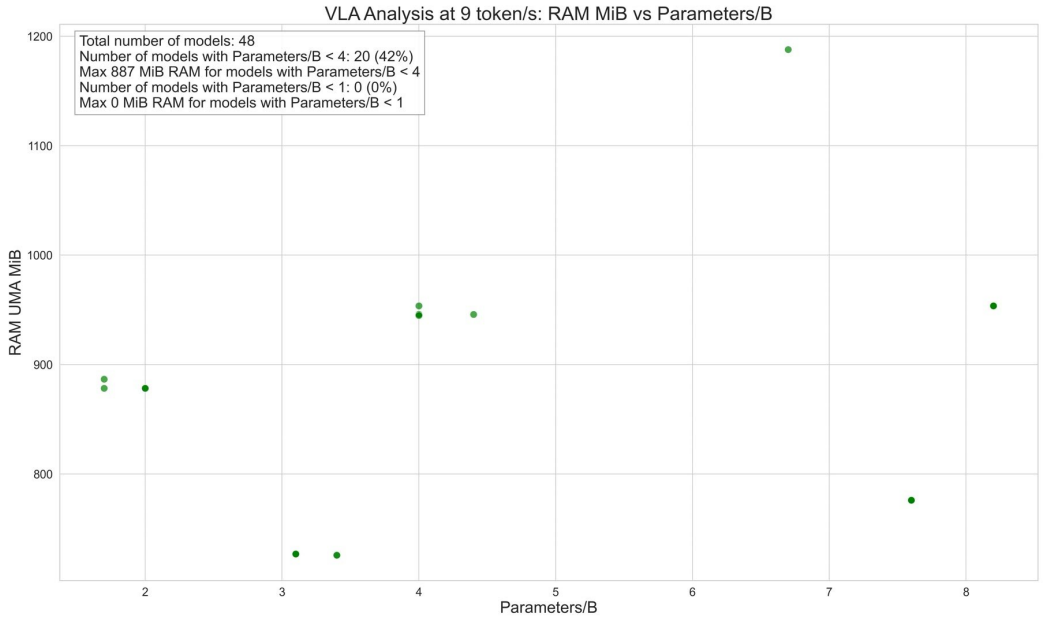

Figure 15. Vision-language-action parameter–UMA scatter for the restricted size range and non-trivial residency of retained VLA checkpoints.

Table 14 is short because the compact VLA branch was represented by only two frontier backbones, qwen3vl and qwen2vl, drawn from a full category with three unique backbone labels. Qwen2VL accounted for 56.25% of retained VLA rows [13,32], and even the lowest listed action-oriented point required 19.329 GBps and 179.956 GFLOPS. The narrow public GGUF coverage makes the contrast exploratory, but the observed floor is still well above the lowest text/audio cases.

Table 14. Backbone-unique vision-language-action frontier for models with below 4 B parameters in the category-specific run.

Model	Backbone	Param./B	GBps	GFLOPS	UMA/MiB
RS4-VLA-2B-GGUF	qwen3vl	2.00	19.329	179.956	878.46
Curious-VLA-GGUF	qwen2vl	3.40	21.195	229.395	725.91

4.8. Cross-Category Mean, Standard-Deviation, and Range Summary

The compute–bandwidth summary in Table 15 provides the clearest cross-category separation. VL and thinking were the heaviest average operating regions in the category-specific regime, with mean bandwidth values of 119.26 and 104.05 GBps and mean compute values of 1287.85 and 1153.98 GFLOPS. VLA formed a smaller elevated multimodal branch at 36.36 GBps and 401.48 GFLOPS, whereas audio defined the lightest overall mean envelope at 13.41 GBps and 134.03 GFLOPS. The compact subset preserved the same qualitative split: conversational, instruct, and audio clustered near 8.5–10.1 GBps and below 101 GFLOPS on average, whereas thinking, VL, and VLA remained higher. The major asymmetry was therefore not memory-only; it appeared most strongly in the resources that determine whether a target generation rate can be met.

Figure 16 turns the same summary into a visual ordering. In bandwidth, audio is lowest, conversational and instruct occupy a lower text-centered band, VLA is intermediate, and thinking and VL form the high-bandwidth region. Compute separates the categories even more strongly: VL and thinking exceed 1100 GFLOPS on average, while audio remains near 134 GFLOPS. The error bars confirm overlap at the individual-model level, but the means still reveal a clear deployment-envelope asymmetry under the category-specific service targets.

Table 15. Cross-category mean, standard-deviation, and range summary for minimum bandwidth and minimum compute in the category-specific run.

Category	Models	Mean GBps	SD GBps	Min GBps	Max GBps	Mean GFLOPS	SD GFLOPS	Min GFLOPS	Max GFLOPS
Conversational	15,205	22.46	10.86	0.03	75.71	252.53	133.67	0.03	656.54
Instruct	581	25.61	11.60	0.06	76.78	304.39	146.16	0.06	656.54
Thinking	3324	104.05	50.82	0.76	253.81	1153.98	627.14	1.01	3021.68
Audio	451	13.41	10.08	0.29	45.86	134.03	119.35	2.60	518.12
Vision-Language	1430	119.26	56.30	8.06	245.27	1287.85	683.95	55.37	3100.78
Vision-Language-Action	48	36.36	14.58	19.33	73.58	401.48	182.60	175.01	933.40

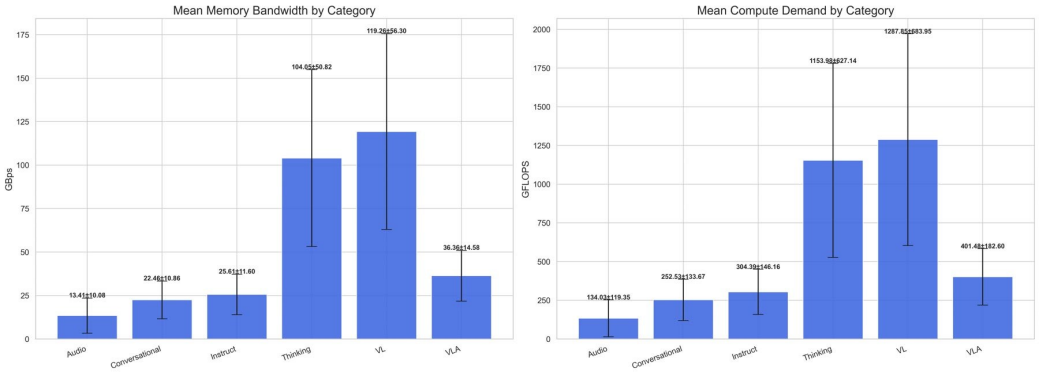

Figure 16. Cross-category distributions of minimum bandwidth and minimum compute under category-specific throughput targets ($n = 21{,}039$). The panels use the same parser-derived GBps and GFLOPS definitions reported in Table 15.

Memory produced a weaker cross-category separation than compute. In Table 16, mean UMA ranged from 725.55 MiB in the instruct branch to 869.46 MiB in the VL branch, whereas the mean compute and bandwidth spread was much wider. The minima were still asymmetric because VL and VLA did not approach the smallest memory floors observed in the text and audio branches. Compact-model availability also differed sharply: audio had 351 of 451 entries below 4 B parameters, while VLA had only 20 of 48 below 4 B and none below 1 B.

Table 16. Cross-category mean, standard-deviation, and range summary for parameter count, UMA, and compact-subset counts in the category-specific run. Parameter minima below 0.01 B are displayed as <0.01.

Category	Models	<4 B	<1 B	Mean Param./B	SD Param./B	Min Param./B	Max Param./B	Mean UMA/MiB	SD UMA/MiB	Min UMA/MiB	Max UMA/MiB
Conversational	15,205	5184	1293	5.55	2.88	<0.01	10.00	815.60	325.53	15.40	2088.96
Instruct	581	133	37	5.93	2.53	0.02	9.20	725.55	397.59	15.94	2088.96
Thinking	3324	1270	221	5.16	2.84	0.02	9.80	829.36	273.00	16.18	2037.76
Audio	451	351	156	2.86	2.44	0.07	9.20	826.00	306.89	14.66	1740.80
Vision-Language	1430	593	99	5.08	2.80	0.10	9.40	869.46	253.12	16.06	1433.60
Vision-Language-Action	48	20	0	5.26	2.44	1.70	8.20	834.77	108.56	725.91	1187.84

Figure 17 reinforces that parameter count alone did not determine memory residency. The category means occupy a relatively narrow band compared with the compute–bandwidth figure, but the within-category dispersion is large, especially for conversational, instruct, audio, and thinking workloads. Figure 18 explains part of this behavior by showing that categories were not sampled from the same parameter-count distribution. Audio was skewed toward smaller checkpoints, VLA lacked the sub-1 B regime, and the text and reasoning categories covered a wider span up to the 10 B cap. Together, the two figures

show why memory must be reported directly instead of inferred from parameter count or throughput target.

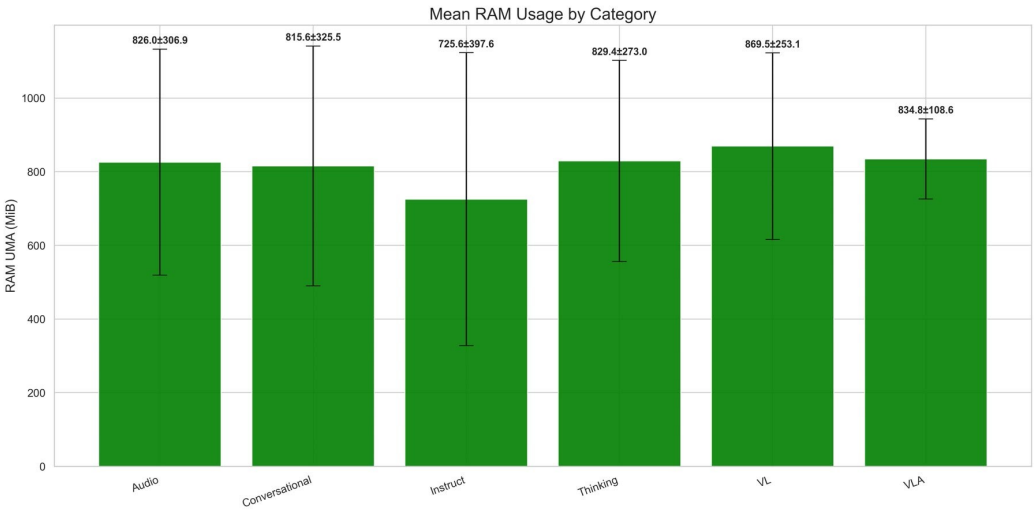

Figure 17. Cross-category distributions of UMA requirements under category-specific throughput targets (n = 21,039). UMA values are parser-estimated backbone-side memory requirements in MiB.

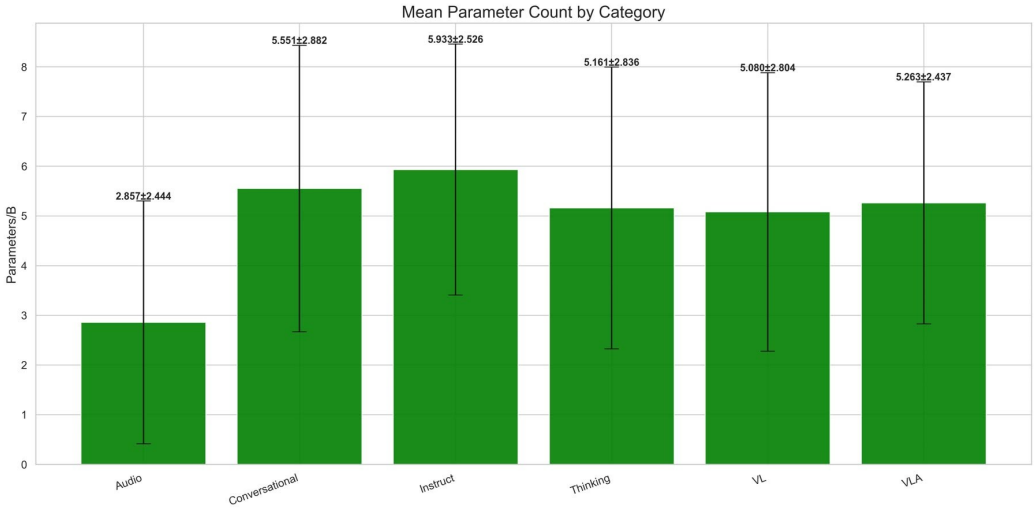

Figure 18. Cross-category parameter-count distributions in the category-specific corpus (n = 21,039). The plot is used as a corpus-composition diagnostic rather than as a substitute for direct compute, bandwidth, or memory estimates.

Table 17 summarizes the backbone-popularity asymmetry that accompanied the hardware asymmetry. The pooled corpus contained 101 unique backbones, but only llama appeared in all six categories. Conversational workloads showed the broadest backbone diversity, whereas VLA showed the narrowest. Llama dominated the text/audio-centric categories, while Qwen2VL dominated both visual branches [13,28]. This composition matters because the category effect is not a pure task-label effect: it also reflects which backbone families have been converted into GGUF artifacts for each workload.

The backbone split in Figure 19 makes the corpus-composition issue explicit. The text and audio branches are dominated by llama-family rows but still retain many secondary backbones [28]. The visual branches are more concentrated around Qwen2VL and related visual backbones [13,32], and the VLA branch is especially narrow. This architecture imbalance is a necessary control on interpretation: the observed hardware asymmetry combines workload effects with the families that are available as public GGUF checkpoints.

Table 17. Backbone diversity and dominant-backbone share across the pooled corpus and the six workload categories in the category-specific run.

Scope	Models	Unique Backbones	Dominant Backbone	Dominant Share
Pooled corpus	21,039	101	llama	45.29%
Conversational	15,205	83	llama	49.56%
Instruct	581	26	llama	76.59%
Thinking	3324	44	llama	36.49%
Audio	451	27	llama	44.79%
Vision-Language	1430	27	qwen2vl	39.79%
Vision-Language-Action	48	3	qwen2vl	56.25%

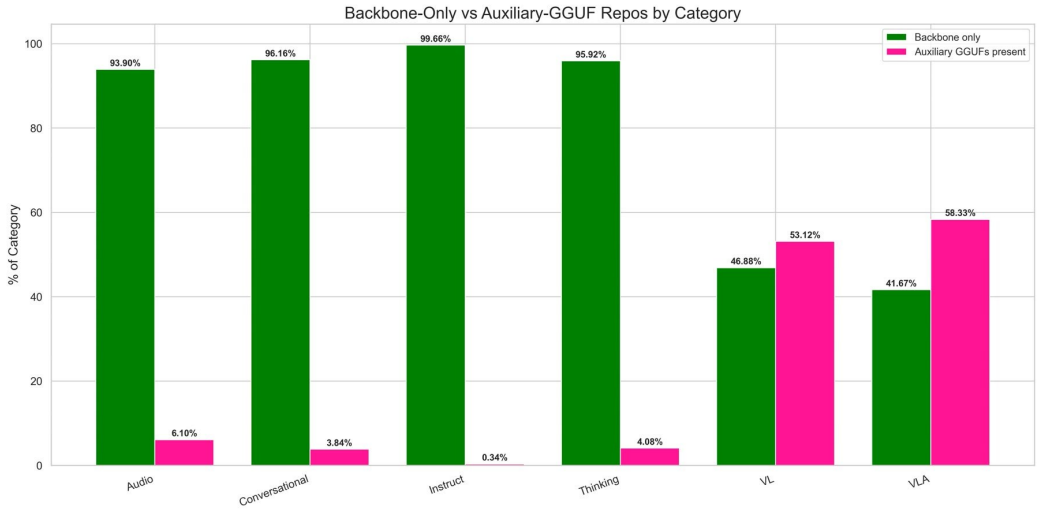

Figure 19. Backbone composition by workload category in the category-specific corpus. The figure reports the retained GGUF backbone labels and highlights that architecture availability differs across workload categories.

4.9. Unified 10-TPS Sensitivity Results

The unified run applied a common 10-TPS target to all categories. It is reported as a sensitivity analysis because it removes the service-level asymmetry of the main category-specific regime while preserving the same model categories and parser-derived metric definitions.

Under the unified 10-TPS target, the cross-category ordering changed substantially. Table 18 no longer places the visual and reasoning branches far above the rest: VL averaged 39.68 GBps and 429.26 GFLOPS, thinking averaged 41.62 GBps and 461.16 GFLOPS, and VLA averaged 40.28 GBps and 445.75 GFLOPS. Instruct became the largest mean compute and bandwidth category, with 608.78 GFLOPS and 51.23 GBps because its original target doubled from 5 to 10 TPS. Audio remained the lightest branch, but the separation was much smaller than in the category-specific regime.

Table 18. Cross-category summary for the unified 10-TPS sensitivity run.

Category	Models	Mean GBps	SD GBps	Mean GFLOPS	SD GFLOPS	Mean UMA/MiB	SD UMA/MiB
Conversational	15,205	44.93	21.72	505.07	267.35	815.60	325.53
Instruct	581	51.23	23.20	608.78	292.32	725.55	397.59
Thinking	3324	41.62	20.35	461.16	250.40	829.36	273.00
Audio	451	26.83	20.15	268.07	238.70	826.00	306.89
Vision-Language	1430	39.68	18.74	429.26	227.75	869.46	253.12
Vision-Language-Action	48	40.28	16.11	445.75	202.01	834.77	108.56

Figure 20 produces the strongest sensitivity result: the large category-specific separation of VL and thinking from the text/audio categories largely collapses, and the means fall within the narrower ranges of approximately 26.83–51.23 GBps and 268.07–608.78 GFLOPS. Category differences remain, but the compression indicates that different target service rates contributed substantially to the main-regime separation. Figure 21 changes much less, so memory residency was comparatively insensitive to the throughput label. Figure 22 and Figure 23 retain their roles as corpus-composition diagnostics because the unified run changes the operating target rather than the underlying public model mix.

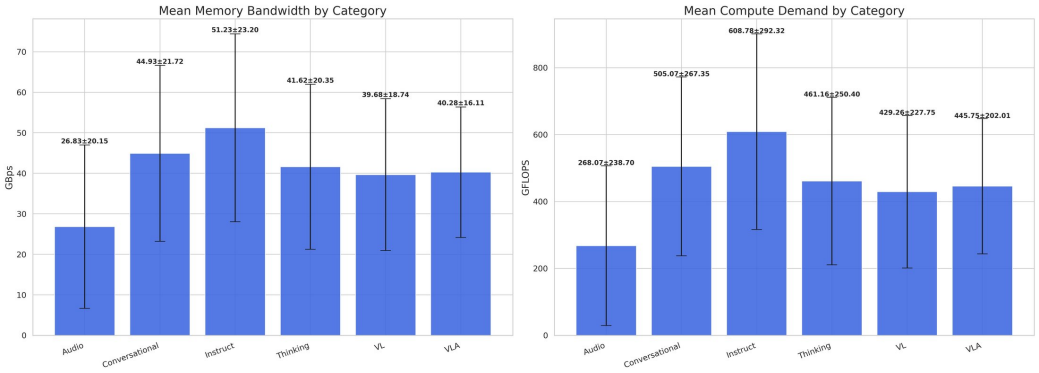

Figure 20. Cross-category minimum bandwidth and minimum compute distributions under the unified 10-TPS target (n = 21,039). The figure isolates service-rate sensitivity by holding the target TPS constant across all workload categories.

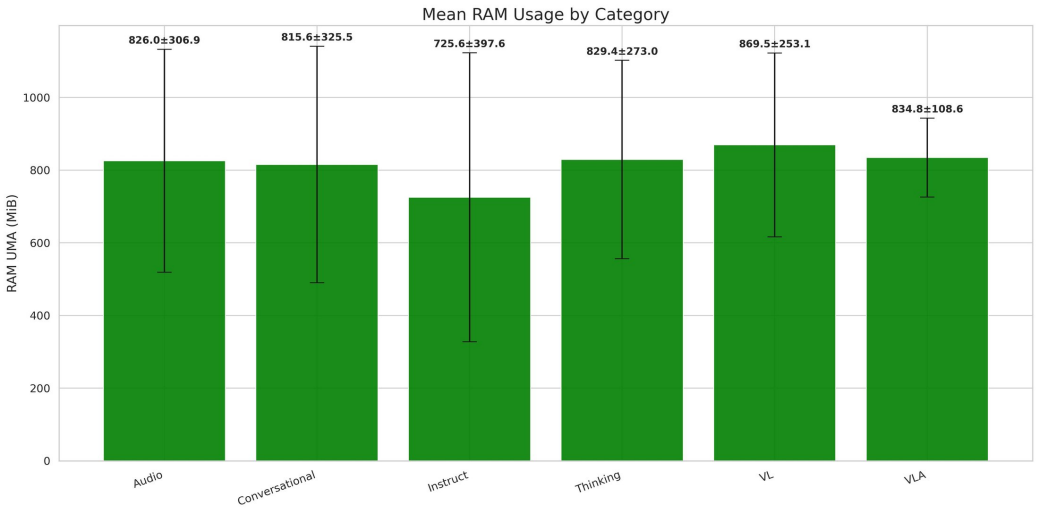

Figure 21. Cross-category UMA distributions under the unified 10-TPS target (n = 21,039). UMA values remain tied to the retained model artifacts and therefore change much less than compute or bandwidth.

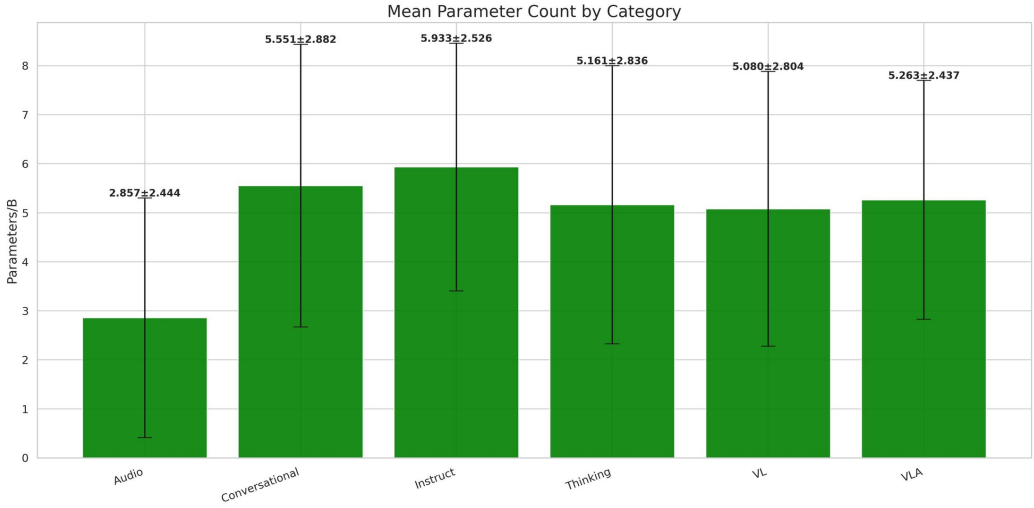

Figure 22. Parameter-count distributions in the unified 10-TPS sensitivity corpus.

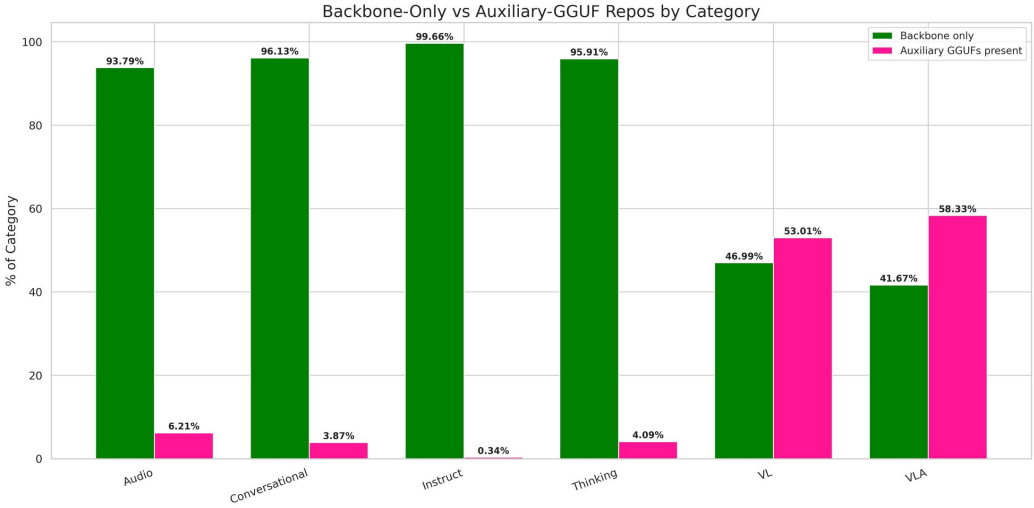

Figure 23. Backbone composition by workload category in the unified 10-TPS sensitivity corpus.

4.10. Inferential Statistical Results

The inferential layer was applied after model discovery, parser-based estimation, and threshold search had produced the deployment-metric exports. After applying the same retained-model identity set to the unified sensitivity run, the category-specific analysis set and the unified 10-TPS analysis set both contained 21,039 rows. In both runs, all rows passed the required numeric and positivity checks for GBps, GFLOPS, and UMA/MiB, so the inferential tables did not depend on post hoc removal of invalid metric values. The statistical script computed all pairwise post hoc contrasts for every estimable subset–metric combination. The selected manuscript-facing rows were chosen to explain the main heavy-versus-light category structure, not to replace the complete exported pairwise table.

All omnibus tests in Table 19 were statistically significant, but the effect magnitudes depended strongly on metric and throughput regime. In the category-specific full corpus, the log-scale Welch summaries were large for minimum bandwidth and minimum compute ($\omega^2_{\log} = 0.44$ and 0.35), and the Kruskal–Wallis robustness checks gave similar rank-based magnitudes ($\epsilon^2 = 0.43$ and 0.35). UMA memory, by contrast, had a small effect magnitude despite its statistical significance ($\omega^2_{\log} = 0.01$; $\epsilon^2 = 0.01$). The below 4 B subset strengthened the compute–bandwidth conclusion, reaching $\omega^2_{\log} = 0.50$ and $\epsilon^2 = 0.48$ for bandwidth and $\omega^2_{\log} = 0.41$ and $\epsilon^2 = 0.43$ for compute.

Table 19. Omnibus inferential summary from the Welch/Games–Howell and Kruskal–Wallis/Dunn–Holm analyses. Welch tests were run on the log scale; Kruskal–Wallis tests were run as nonparametric robustness checks.

Regime	Subset	Metric	Welch F	Welch p	$\omega^2_{\log}$	Kruskal H	Kruskal p	ϵ^2
Category-specific TPS	All	Minimum bandwidth	3693.15	<0.001	0.44	8966.46	<0.001	0.43
Category-specific TPS	All	Minimum compute	2532.64	<0.001	0.35	7320.44	<0.001	0.35
Category-specific TPS	All	UMA memory	39.47	<0.001	0.01	144.39	<0.001	0.01
Category-specific TPS	<4 B	Minimum bandwidth	2083.04	<0.001	0.50	3661.84	<0.001	0.48
Category-specific TPS	<4 B	Minimum compute	1423.77	<0.001	0.41	3245.85	<0.001	0.43
Category-specific TPS	<4 B	UMA memory	18.87	<0.001	0.01	81.03	<0.001	0.01
Unified 10 TPS	All	Minimum bandwidth	52.76	<0.001	0.02	463.43	<0.001	0.02
Unified 10 TPS	All	Minimum compute	59.22	<0.001	0.02	629.63	<0.001	0.03
Unified 10 TPS	All	UMA memory	39.47	<0.001	0.01	144.39	<0.001	0.01
Unified 10 TPS	<4 B	Minimum bandwidth	175.61	<0.001	0.01	26.47	<0.001	0.00
Unified 10 TPS	<4 B	Minimum compute	89.43	<0.001	0.01	52.41	<0.001	0.01
Unified 10 TPS	<4 B	UMA memory	18.87	<0.001	0.01	81.03	<0.001	0.01

The unified 10-TPS results changed the interpretation. Minimum bandwidth and compute remained statistically significant, but their effect sizes dropped to $\omega^2_{\log} = 0.02$ and 0.02 in the full corpus, with Kruskal–Wallis $\epsilon^2 = 0.02$ and 0.03. In the 4 B subset below, the unified bandwidth and compute effects were smaller still on a nonparametric scale, with $\epsilon^2 = 0.00$ and 0.01 after rounding. This compression indicates that much of the strong compute–bandwidth separation in the main regime came from the deliberately different category-specific service targets. The memory effect remained small and close to the category-specific estimate, supporting the view that memory asymmetry was driven more by model structure and backbone mix than by throughput target.

The selected manuscript-facing contrasts in Table 20 clarify the source of the omnibus effects while the full statistical output contains the all-pairs post hoc tables. Under category-specific targets, VL required much larger geometric-mean bandwidth and compute than audio: the ratios were 10.94 and 13.34, respectively. Thinking required 4.71 times the geometric-mean bandwidth of conversational models, and VLA required 1.90 times the geometric-mean compute of conversational models in the full corpus. In the below 4 B subset, the reported visual/action contrasts remained strong: VL required 6.79 times the bandwidth of conversational models, and VLA required 3.18 times the compute. Under the unified 10-TPS target, these ratios compressed: VL remained above audio, but VLA no longer differed meaningfully from conversational models in full-corpus bandwidth. The Dunn–Holm robustness analyses supported the same interpretation. Under category-specific targets, rank dominance was large for heavy visual/reasoning categories against light categories, whereas under the unified compact subset many corresponding rank effects became negligible or small even when some adjusted probability values remained below 0.05. This is why the probability values are interpreted together with ratios, confidence intervals, and nonparametric effect magnitudes rather than in isolation.

Table 20. Selected manuscript-facing Games–Howell geometric-mean contrasts from the statistical analyses. Ratios above one indicate that the first category required the larger geometric-mean deployment metric. The statistical output contains all pairwise post hoc contrasts; this table reports selected rows for interpretation.

Regime	Subset	Contrast	Metric	Ratio [95% CI]	*p*	Sig.
Category-specific TPS	All	VL vs. Audio	Minimum bandwidth	10.94 [9.49, 12.60]	<0.001	Yes
Category-specific TPS	All	VL vs. Audio	Minimum compute	13.34 [11.20, 15.88]	<0.001	Yes
Category-specific TPS	All	Thinking vs. Conversational	Minimum bandwidth	4.71 [4.52, 4.90]	<0.001	Yes
Category-specific TPS	All	VLA vs. Conversational	Minimum compute	1.90 [1.55, 2.33]	<0.001	Yes
Category-specific TPS	<4 B	VL vs. Conversational	Minimum bandwidth	6.79 [6.28, 7.34]	<0.001	Yes
Category-specific TPS	<4 B	VLA vs. Conversational	Minimum compute	3.18 [2.92, 3.47]	<0.001	Yes
Unified 10 TPS	All	VL vs. Audio	Minimum bandwidth	1.82 [1.58, 2.10]	<0.001	Yes
Unified 10 TPS	All	VLA vs. Conversational	Minimum bandwidth	1.02 [0.85, 1.21]	1.000	No
Unified 10 TPS	<4 B	VLA vs. Conversational	Minimum compute	1.77 [1.62, 1.93]	<0.001	Yes

5. Discussion

The central result of the updated analysis is a symmetry–asymmetry contrast. Discovery, parsing, resource minimization, context size, and summary definitions were kept symmetric across the retained categories, yet the category-specific throughput regime produced strongly asymmetric hardware envelopes. The asymmetry was largest on the compute–bandwidth axis: VL and thinking workloads were the heaviest on average, VLA formed a smaller but still elevated multimodal branch, and audio, instruct, and conversational workloads remained lighter.

The statistical layer qualifies that conclusion in an important way. When the category-specific TPS targets were preserved, both the Welch and Kruskal–Wallis analyses showed large category effects for minimum bandwidth and minimum compute. When all categories were reinterpreted under the unified 10-TPS sensitivity run, the same omnibus tests remained significant, but the effect magnitudes became much smaller. Therefore, cross-category differences should not be attributed only to intrinsic modality or backbone structure; part of the observed separation is a consequence of the target service rate assigned to each workload. This is not a flaw in the main scenario because different edge services can legitimately target different token rates, but it should be made explicit when comparing categories.

Memory behaved differently from bandwidth and compute. UMA differences were statistically detectable, but the effect magnitudes were small in both the category-specific and unified analyses. This finding is consistent with the descriptive tables: mean UMA changed within a relatively narrow band across categories, while bandwidth and compute moved much more strongly. In deployment terms, parameter count and throughput target were not sufficient to predict memory residency. Sidecar components, tokenizer and projection structure, backbone family, cache demand, and parser-estimated activation buffers all remain relevant to memory budgeting.

The auxiliary VL context-window analysis sharpened this point. The 128k-context subset had a lower median parameter count than the 2k-context VL subset, yet its median UMA memory was more than twice as high. This inversion shows that context allocation and cache-related buffers can dominate model-count intuition: a smaller checkpoint configured for a long context can be less memory-friendly than a larger checkpoint configured for a short context. The context-window figures therefore extend the main memory

conclusion from cross-category comparison to an intra-category deployment choice: the same workload family can change its memory envelope substantially when the context target changes.

Backbone statistics sharpen the same asymmetry from a different angle. The category-specific corpus contained 101 unique backbone labels, but that diversity was unevenly distributed. Conversational workloads had the broadest architecture mix, whereas VLA was concentrated into only a few visual backbones. The dominant families were therefore category-dependent rather than uniformly distributed across the corpus [13,28]. Consequently, the category label captured a mixture of workload mode, target service rate, and the model families that were available in GGUF form.

Several limitations should temper interpretation. First, the corpus is subject to GGUF availability bias: models that were not publicly converted to GGUF, not released with parser-readable metadata, distributed only through framework-native checkpoints, or served only behind APIs could not enter the analysis. The findings therefore characterize the public Q4-K-M GGUF deployment ecosystem, not the entire population of generative models. Second, the reported values are parser-derived deployment estimates rather than direct measurements on fixed hardware. Third, the main regime intentionally used category-specific target TPS values, and the unified 10-TPS run should therefore be read as a sensitivity analysis rather than as a replacement for the deployment-scenario analysis. Fourth, multimodal sidecar components such as `mmproj` files were not represented uniformly across repositories, so the reported metrics are best interpreted as backbone-side estimates. Fifth, the VLA category remained small, and its reported contrast estimates are correspondingly less stable than those for the much larger conversational and thinking categories. Finally, statistical significance is easy to obtain in the largest subsets; effect sizes and confidence intervals are more informative than probability values alone.

Future work should incorporate multimodal sidecar modules, perception encoders, audio front ends, and action heads into end-to-end system budgets. The same workflow should also be calibrated against direct board-level measurements and paired with task-quality measures so that edge deployment can be evaluated through explicit accuracy–latency–bandwidth–memory trade-offs rather than through parser-derived feasibility alone.

6. Conclusions

A symmetric GGUF-based profiling workflow was applied to conversational, instruct, thinking, audio, vision-language, and vision-language-action workloads. In the category-specific run, the retained corpus comprised 21,039 profiled entries and 101 unique backbone labels. The descriptive tables and cross-category figures showed that the strongest asymmetry appeared on the compute–bandwidth axis: VL and thinking workloads were the heaviest on average, VLA occupied a smaller but still elevated multimodal regime, and audio defined the lightest overall mean envelope. In the compact below 4 B parameter subset, conversational, instruct, and audio workloads remained close to the low-resource cluster, whereas thinking, VL, and VLA retained substantially larger bandwidth and compute requirements. The auxiliary VL context-window comparison further showed that long-context operation can increase memory residency even when the retained checkpoint set is smaller on average.

The inferential results support the descriptive findings while clarifying their scope. Category-specific target TPS values produced large bandwidth and compute effects, but the unified 10-TPS sensitivity run compressed those effects substantially. Memory differences were statistically detectable but small in magnitude. The resulting conclusion is that a common measurement procedure does not imply a common hardware envelope: deployment feasibility depends jointly on workload category, throughput target, backbone family,

and explicit memory budgeting. The practical implication is that edge-model screening should report compute, bandwidth, and memory together, and should state the assumed service rate explicitly. Future work should extend the analysis from backbone-side GGUF estimates to end-to-end multimodal systems and direct hardware measurements.

Author Contributions: Conceptualization, N.K. and D.P.P.; methodology, N.K. and D.P.P.; software, N.K.; validation, N.K. and D.P.P.; formal analysis, N.K. and D.P.P.; investigation, N.K. and D.P.P.; resources, D.P.P.; data curation, N.K.; writing—original draft preparation, N.K. and D.P.P.; writing—review and editing, N.K. and D.P.P.; visualization, N.K.; supervision, D.P.P.; project administration, D.P.P.; funding acquisition, D.P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Public model assets referenced in this study are available from the official Hugging Face, GitHub, and Figure pages cited in the references. The derived metric tables, visualization assets, and analysis scripts used to generate the reported results, tables, and plots will be made available upon publication of this article.

Conflicts of Interest: Authors Danilo Pietro Pau and Niks Kordjukovs were employed by the System Research and Applications, STMicroelectronics, Agrate Brianza, Italy. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Unified 10 TPS Per-Category Diagnostic Figures

The appendix complements the unified 10-TPS summary in Table 18 and Figures 20–23 with per-category diagnostic views. The same 21,039 retained model identities were used in the category-specific and unified regimes, so the sensitivity comparison is not driven by a change in sample membership. The compute–bandwidth panels should be read as service-rate diagnostics, whereas the parameter–UMA panels serve as checks on model residency and corpus composition.

For conversational models, the unified target doubles the original 5-TPS scenario. Figures A1 and A2 therefore move the compute–bandwidth envelope upward while leaving the broad residency scatter largely intact. The contrast between the shifted throughput panel and the diffuse UMA panel reinforces that target rate mainly changes the operating point, not the underlying memory dispersion.

The instruct diagnostic pair in Figures A3 and A4 explains the ranking change in Table 18. Moving from the category-specific 5-TPS target to 10 TPS raises the required compute and bandwidth for the same retained entries, which is why instruct becomes the largest mean category in that sensitivity run. Its UMA scatter does not simplify in parallel: low-parameter entries still span both low and high memory requirements.

Thinking moves in the opposite direction from the categories originally evaluated at 5 TPS. In Figures A5 and A6, reducing the target from 25 to 10 TPS compresses the compute–bandwidth plane and brings the reasoning branch closer to the rest of the unified comparison. The parameter–UMA view still spans a wide range, so the sensitivity mainly affects throughput-side demand.

For audio, the 10-TPS sensitivity run raises the target relative to the main 5-TPS case without changing the category's overall position. Figures A7 and A8 retain a low average compute–bandwidth profile compared with visual and reasoning workloads, even though the envelope shifts upward. The UMA panel prevents a stronger claim: low compute demand and low memory demand are not equivalent for audio checkpoints.

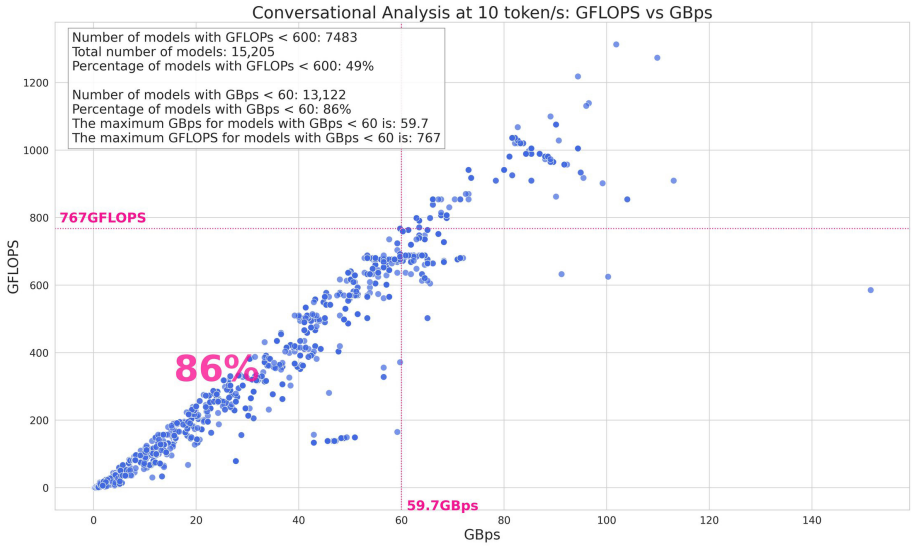

Figure A1. Conversational envelope after raising the target from 5 to 10 TPS in the unified sensitivity run.

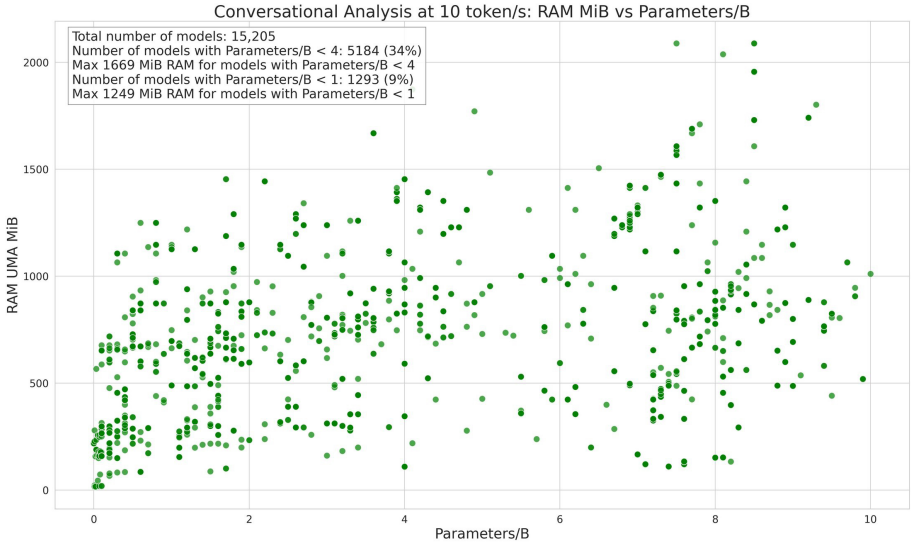

Figure A2. Conversational UMA scatter under the unified target, retained as a residency-composition diagnostic.

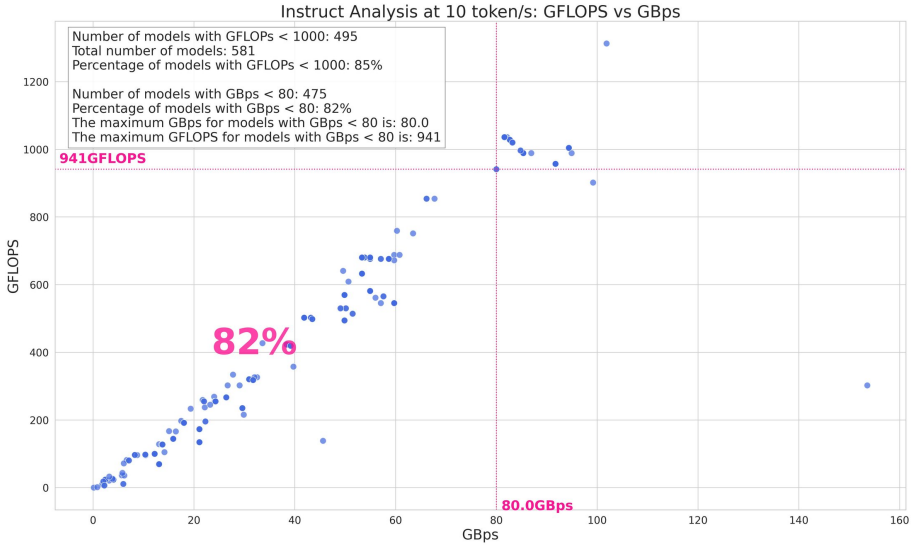

Figure A3. Instruct compute–bandwidth envelope after the 5-to-10-TPS target increase.

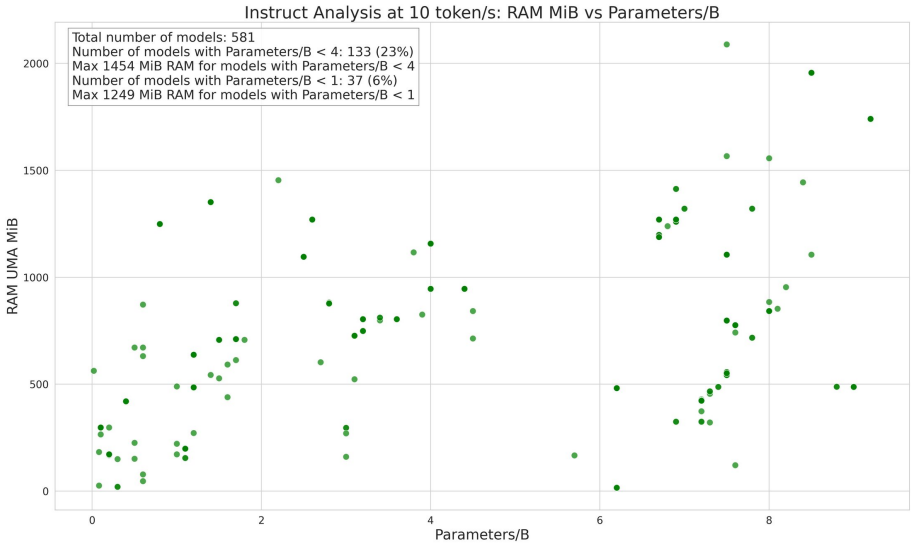

Figure A4. Instruct UMA scatter in the unified run, with memory variation persisting after rate normalization.

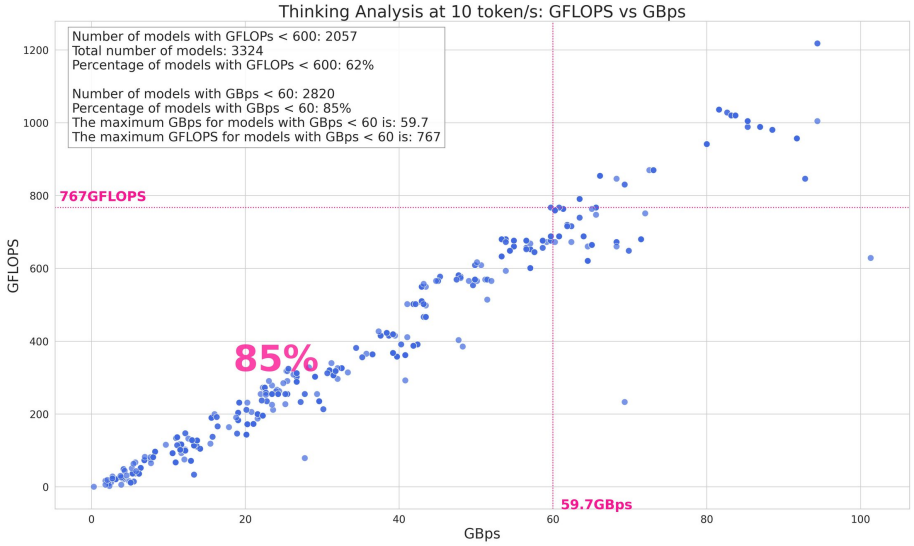

Figure A5. Thinking compute–bandwidth envelope after lowering the target from 25 to 10 TPS.

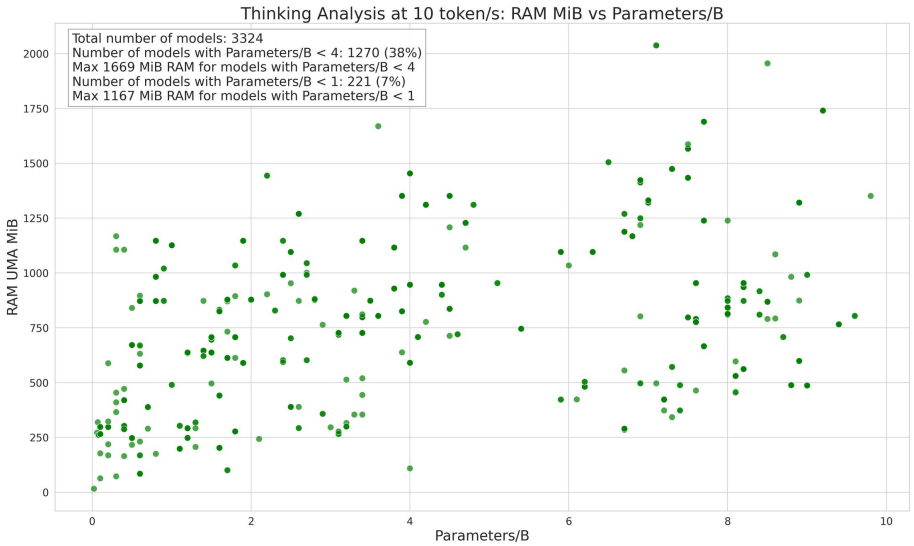

Figure A6. Thinking UMA scatter under the unified target, contrasting rate compression with stable residency variation.

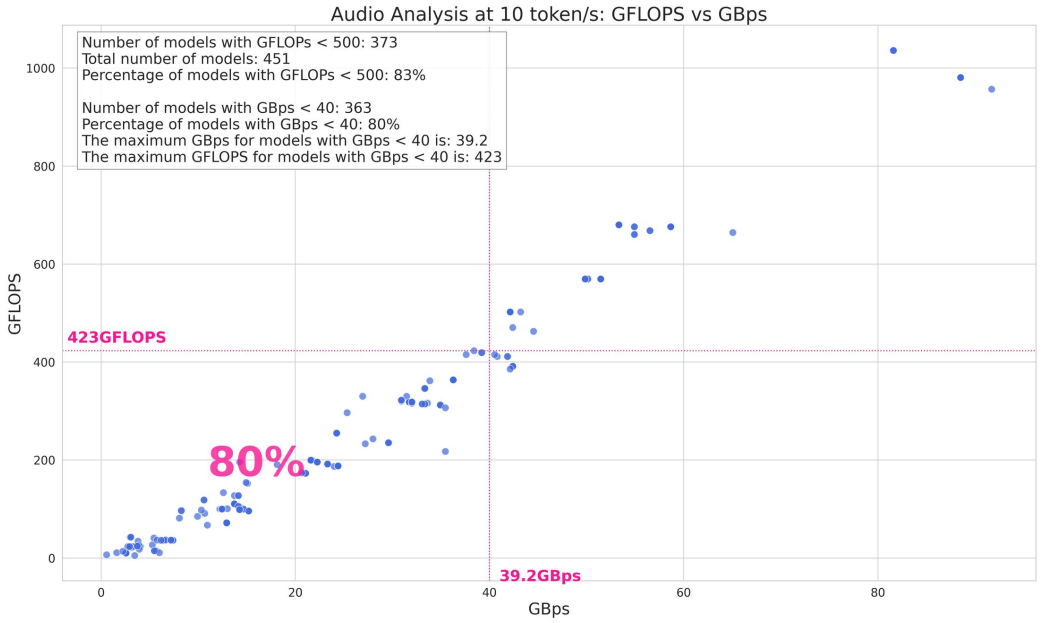

Figure A7. Audio compute–bandwidth envelope after the 5-to-10-TPS sensitivity increase.

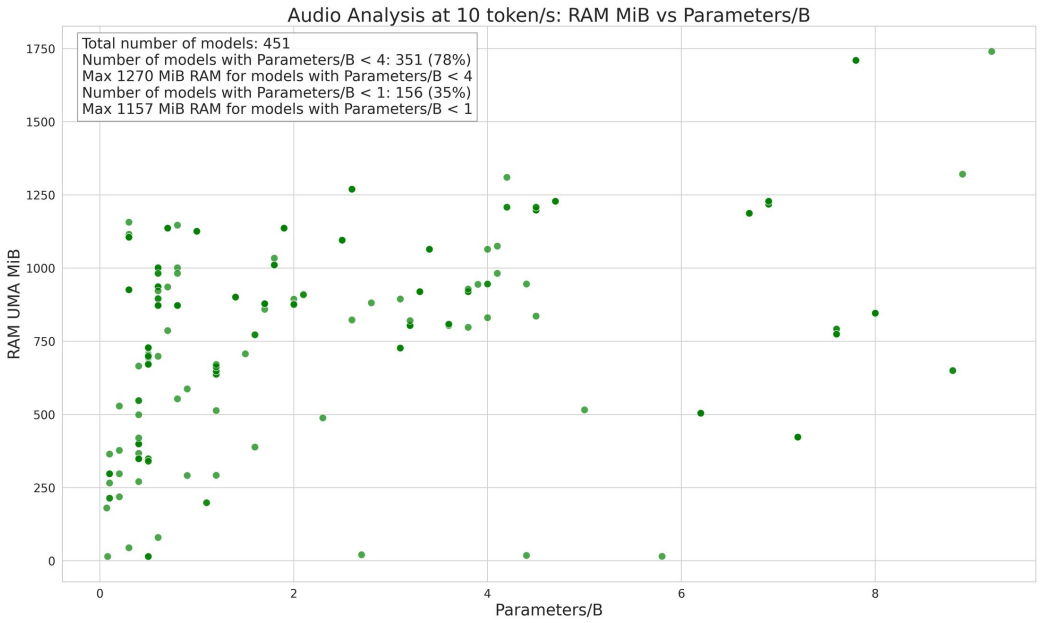

Figure A8. Audio UMA scatter in the unified run, separating low compute demand from memory residency.

The VL diagnostics illustrate the strongest downward adjustment caused by the unified target. Figures A9 and A10 reduce the original 30-TPS scenario to 10 TPS, sharply shrinking the compute–bandwidth envelope and removing VL's dominance of the mean comparison. Memory residency remains comparatively high, so throughput normalization weakens the compute result more than the visual-branch memory result.

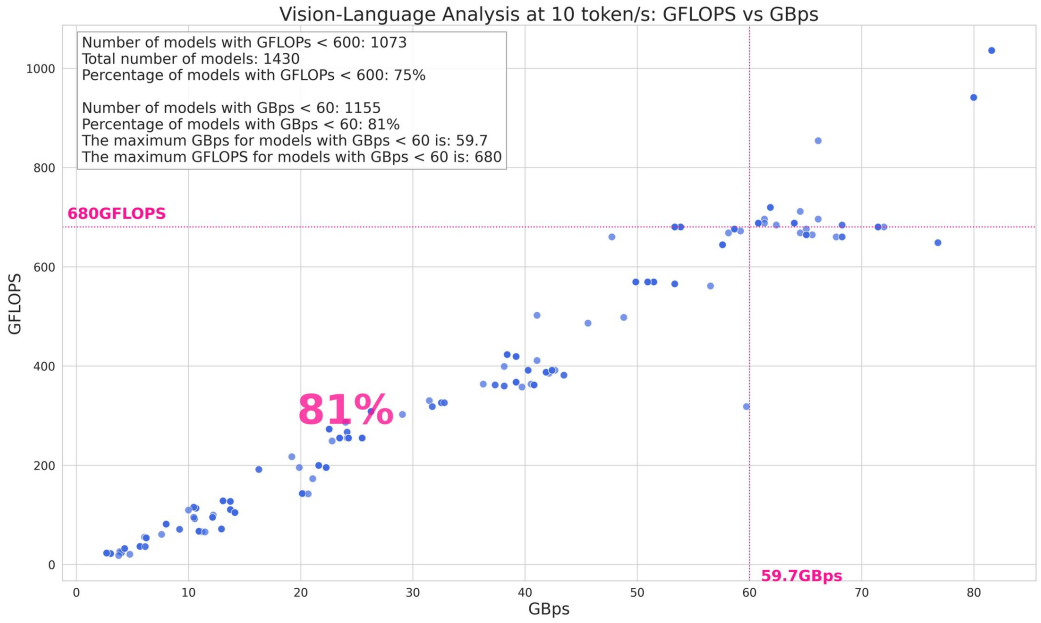

Figure A9. Vision-language compute–bandwidth envelope after reducing the 30-TPS target to 10 TPS.

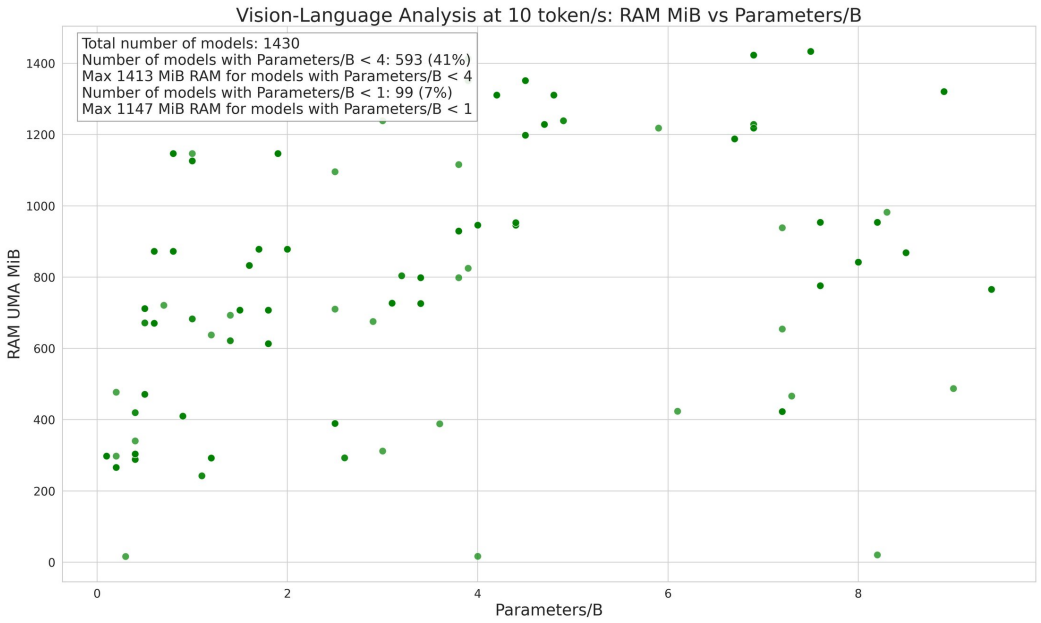

Figure A10. Vision-language UMA scatter under the unified target, retaining the visual-branch memory profile.

Because the VLA main target was already close to 10 TPS, Figures A11 and A12 change less dramatically than the thinking or VL diagnostics. The branch remains small in sample size and elevated relative to the lightest categories. Its memory interpretation is also constrained by public GGUF availability: the retained VLA checkpoints are concentrated in a few visual-action families, so the panel describes the observed branch rather than a comprehensive population.

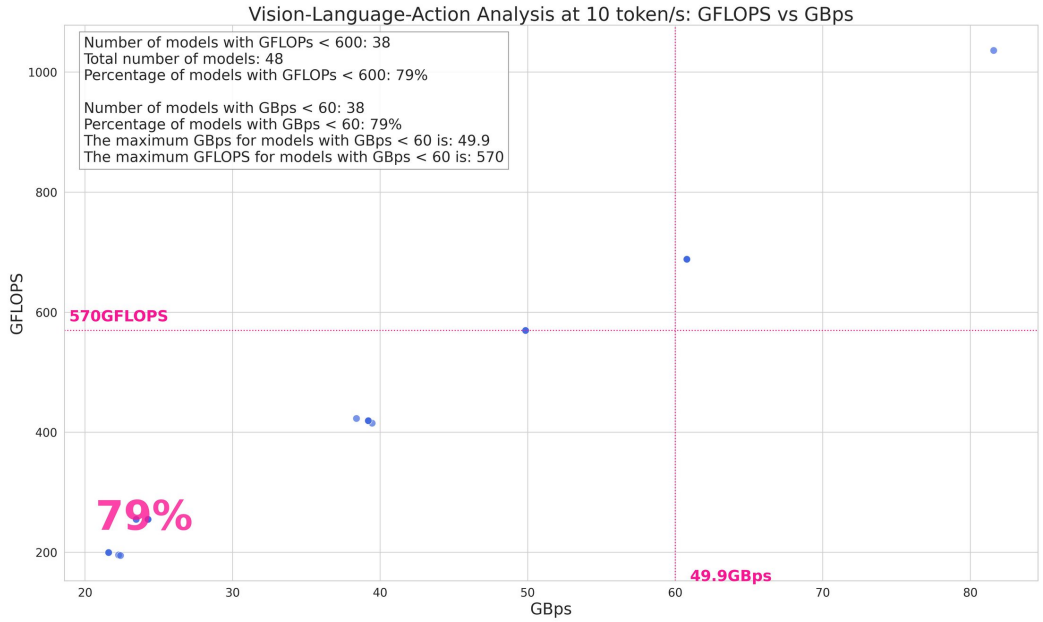

Figure A11. Vision-language-action compute–bandwidth envelope after the small 9-to-10-TPS target adjustment.

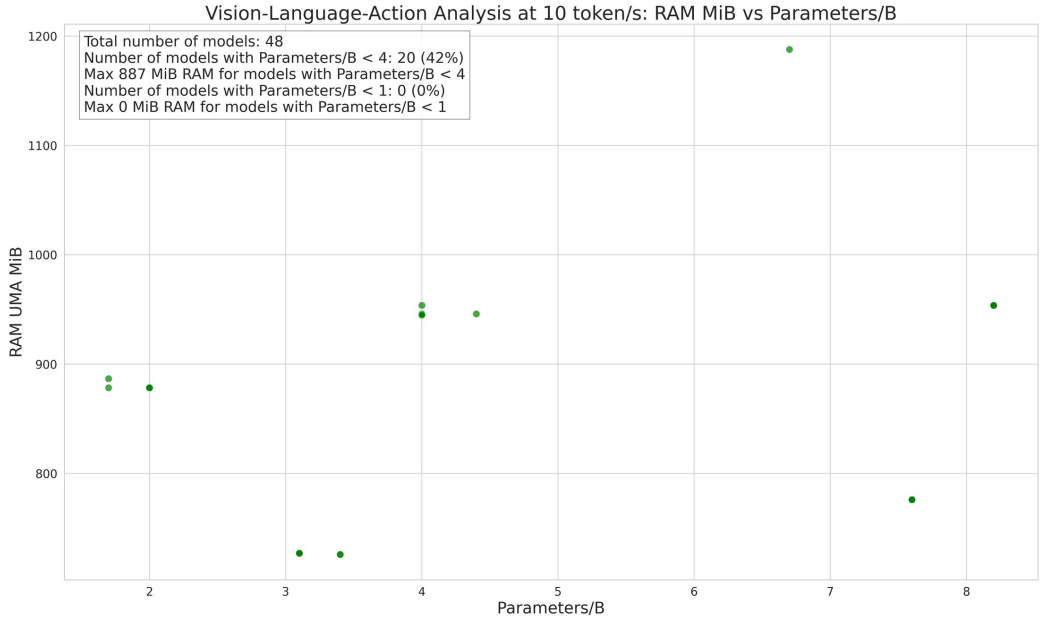

Figure A12. Vision-language-action UMA scatter in the unified run for the sparse visual-action branch.

Appendix B. Online Resources

Appendix links are separated into tooling and implementation resources, followed by one breakable model-resource table per workload category. Table A1 lists the general tooling, documentation, and rule-documentation resources, whereas Tables A2–A7 list the model-resource links by workload category. Model-family links are assigned to the workload categories in which the corresponding backbone label appears in the retained corpus; a link can therefore appear in more than one category table. Rows that were moved only to the appendix are kept as plain URL or DOI links, while rows already cited in the manuscript retain their citation markers. Existing URLs were last accessed on 24 April 2026; newly added backbone rows were checked on 28 April 2026.

Table A1. General online resources and implementation documentation used in this study.

Resource	URL or DOI
SparQ Attention [6]	https://proceedings.mlr.press/v235/ribar24a.html (accessed on 24 April 2026)
GGUF Hub Documentation [10]	https://huggingface.co/docs/hub/gguf (accessed on 24 April 2026)
llama.cpp [11]	https://github.com/ggml-org/llama.cpp (accessed on 24 April 2026)
gguf-parser-go [12]	https://github.com/gpustack/gguf-parser-go (accessed on 24 April 2026)
HF-Agents CLI [15]	https://github.com/huggingface/hf-agents/tree/main (accessed on 24 April 2026)
GGUF Specification [16]	https://github.com/ggml-org/ggml/blob/master/docs/gguf.md (accessed on 24 April 2026)
statsmodels one-way ANOVA documentation [18]	https://www.statsmodels.org/dev/generated/statsmodels.stats.oneway.anova_oneway.html (accessed on 24 April 2026)
Classification rule documentation repository	https://github.com/hobbitlv1/complexity_analysis_for_categorized_edge_language_models (accessed on 24 April 2026)
Classification-regex documentation	https://raw.githubusercontent.com/hobbitlv1/complexity_analysis_for_categorized_edge_language_models/main/classification_regexes.txt (accessed on 24 April 2026)
Pipeline/task descriptor documentation	https://raw.githubusercontent.com/hobbitlv1/complexity_analysis_for_categorized_edge_language_models/main/pipeline_tags.txt (accessed on 24 April 2026)
Search-term documentation	https://raw.githubusercontent.com/hobbitlv1/complexity_analysis_for_categorized_edge_language_models/main/search_terms.txt (accessed on 24 April 2026)

Table A2. Conversational model-resource links.

Resource	URL or DOI
Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution [13]	https://doi.org/10.48550/arXiv.2409.12191 (accessed on 24 April 2026)
Granite 3.0 [25]	https://github.com/ibm-granite/granite-3.0-language-models/blob/main/paper.pdf (accessed on 24 April 2026)
Granite 3.0 MoE [25]	https://github.com/ibm-granite/granite-3.0-language-models/blob/main/paper.pdf (accessed on 24 April 2026)
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [26]	https://doi.org/10.18653/v1/N19-1423 (accessed on 24 April 2026)
Mamba [27]	https://openreview.net/forum?id=tEYskw1VY2 (accessed on 24 April 2026)

Table A2. *Cont.*

Resource	URL or DOI
The Llama 3 Herd of Models [28]	https://doi.org/10.48550/arXiv.2407.21783 (accessed on 24 April 2026)
Qwen3 Technical Report [30]	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
Qwen3-VL Technical Report [32]	https://doi.org/10.48550/arXiv.2511.21631 (accessed on 24 April 2026)
GPT-2 [33]	https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 24 April 2026)
T5 [34]	https://www.jmlr.org/papers/v21/20-074.html (accessed on 24 April 2026)
AFM-4.5B [35]	https://www.arcee.ai/blog/deep-dive-afm-4-5b-the-first-arcee-foundational-model (accessed on 24 April 2026)
Command R [36]	https://huggingface.co/CohereLabs/c4ai-command-r-v01 (accessed on 24 April 2026)
DeciLM 6B [37]	https://huggingface.co/Deci/DeciLM-6b (accessed on 24 April 2026)
ERNIE 4.5 [38]	https://yiyan.baidu.com/blog/publication/ERNIE_Technical_Report.pdf (accessed on 24 April 2026)
Gemma 3n [39]	https://ai.google.dev/gemma/docs/gemma-3n (accessed on 24 April 2026)
Granite 4.0 Hybrid [40]	https://www.ibm.com/new/announcements/ibm-granite-4-0-hyper-efficient-high-performance-hybrid-models (accessed on 24 April 2026)
Jais 2 [41]	https://www.cerebras.ai/blog/jais2 (accessed on 24 April 2026)
Llama 4 [42]	https://ai.meta.com/blog/llama-4-multimodal-intelligence/ (accessed on 24 April 2026)
Maincoder-1B [43]	https://huggingface.co/Maincode/Maincoder-1B (accessed on 24 April 2026)
MosaicBERT [44]	https://openreview.net/forum?id=5zipcfLC2Z (accessed on 24 April 2026)
Persimmon-8B [45]	https://www.adept.ai/blog/persimmon-8b/ (accessed on 24 April 2026)
Phi-2 [46]	https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ (accessed on 24 April 2026)
PLaMo 3 [47]	https://tech.preferred.jp/ja/blog/plamo_3_8b_31b/ (accessed on 24 April 2026)
Qwen3.5 [48]	https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026)
Rnj-1 [49]	https://essential.ai/research/rnj-1 (accessed on 24 April 2026)

Table A2. *Cont.*

Resource	URL or DOI
SmolLM3 [50]	https://huggingface.co/blog/smollm3 (accessed on 24 April 2026)
XVERSE-7B [51]	https://huggingface.co/xverse/XVERSE-7B (accessed on 24 April 2026)
Apertus: Democratizing Open and Compliant LLMs for Global Language Environments	https://doi.org/10.48550/arXiv.2509.14233 (accessed on 24 April 2026)
Arcee Trinity Large Technical Report	https://doi.org/10.48550/arXiv.2602.17004 (accessed on 24 April 2026)
ARWKV: Pretrain is not what we need, an RNN-Attention-Based Language Model Born from Transformer	https://doi.org/10.48550/arXiv.2501.15570 (accessed on 24 April 2026)
AXL Architecture eXperimental Lab documentation (`multiscale_transformer`)	https://huggingface.co/datasets/KoinicLabs/AXL-Architecture-eXperimental-Lab (accessed on 24 April 2026)
Baichuan 2: Open Large-scale Language Models	https://doi.org/10.48550/arXiv.2309.10305 (accessed on 24 April 2026)
Bailing-MoE-V2 architecture documentation (`bailingmoe2`)	https://huggingface.co/tiny-random/bailing-moe-v2 (accessed on 24 April 2026)
BLOOM: A 176B-Parameter Open-Access Multilingual Language Model	https://doi.org/10.48550/arXiv.2211.05100 (accessed on 24 April 2026)
ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools	https://doi.org/10.48550/arXiv.2406.12793 (accessed on 24 April 2026)
CodeShell Technical Report	https://doi.org/10.48550/arXiv.2403.15747 (accessed on 24 April 2026)
Command A: An Enterprise-Ready Large Language Model	https://doi.org/10.48550/arXiv.2504.00698 (accessed on 24 April 2026)
DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model	https://doi.org/10.48550/arXiv.2405.04434 (accessed on 24 April 2026)
Dream 7B: Diffusion Large Language Models	https://doi.org/10.48550/arXiv.2508.15487 (accessed on 24 April 2026)
Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence	https://doi.org/10.48550/arXiv.2404.05892 (accessed on 24 April 2026)
EuroBERT: Scaling Multilingual Encoders for European Languages	https://doi.org/10.48550/arXiv.2503.05500 (accessed on 24 April 2026)
EXAONE 3.5: Series of Large Language Models for Real-world Use Cases	https://doi.org/10.48550/arXiv.2412.04862 (accessed on 24 April 2026)

Table A2. *Cont.*

Resource	URL or DOI
EXAONE 4.0 Technical Report	https://doi.org/10.48550/arXiv.2507.11407 (accessed on 24 April 2026)
Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance	https://doi.org/10.48550/arXiv.2507.22448 (accessed on 24 April 2026)
Gemma 2: Improving Open Language Models at a Practical Size	https://doi.org/10.48550/arXiv.2408.00118 (accessed on 24 April 2026)
Gemma 3 Technical Report	https://doi.org/10.48550/arXiv.2503.19786 (accessed on 24 April 2026)
Gemma 4 model card (`gemma4`)	https://huggingface.co/google/gemma-4-E2B-it (accessed on 24 April 2026)
Gemma: Open Models Based on Gemini Research and Technology	https://doi.org/10.48550/arXiv.2403.08295 (accessed on 24 April 2026)
GPT-NeoX-20B: An Open-Source Autoregressive Language Model	https://doi.org/10.18653/v1/2022.bigscience-1.9 (accessed on 24 April 2026)
gpt-oss-120b/gpt-oss-20b Model Card	https://doi.org/10.48550/arXiv.2508.10925 (accessed on 24 April 2026)
Hunyuan-Large: An Open-Source MoE Model with 52 Billion Activated Parameters	https://doi.org/10.48550/arXiv.2411.02265 (accessed on 24 April 2026)
InternLM2 Technical Report	https://doi.org/10.48550/arXiv.2403.17297 (accessed on 24 April 2026)
Jais and Jais-chat: Arabic-Centric Foundation and Instruction-Tuned Open Generative Large Language Models	https://doi.org/10.48550/arXiv.2308.16149 (accessed on 24 April 2026)
Jamba: A Hybrid Transformer-Mamba Language Model	https://doi.org/10.48550/arXiv.2403.19887 (accessed on 24 April 2026)
Large Language Diffusion Models	https://doi.org/10.48550/arXiv.2502.09992 (accessed on 24 April 2026)
LFM2 Technical Report	https://doi.org/10.48550/arXiv.2511.23404 (accessed on 24 April 2026)
LLaDA-MoE: A Sparse MoE Diffusion Language Model (`llada-moe`)	https://doi.org/10.48550/arXiv.2509.24389 (accessed on 24 April 2026)
Llama 2: Open Foundation and Fine-Tuned Chat Models	https://doi.org/10.48550/arXiv.2307.09288 (accessed on 24 April 2026)
MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies	https://doi.org/10.48550/arXiv.2404.06395 (accessed on 24 April 2026)
Ministral 3	https://doi.org/10.48550/arXiv.2601.08584 (accessed on 24 April 2026)
Nemotron-4 340B Technical Report	https://doi.org/10.48550/arXiv.2406.11704 (accessed on 24 April 2026)

Table A2. *Cont.*

Resource	URL or DOI
Nemotron-H: A Family of Accurate and Efficient Hybrid Mamba-Transformer Models	https://doi.org/10.48550/arXiv.2504.03624 (accessed on 24 April 2026)
OLMo 2: The Best Fully Open Language Model to Date	https://doi.org/10.48550/arXiv.2501.00656 (accessed on 24 April 2026)
OLMo: Accelerating the Science of Language Models	https://doi.org/10.48550/arXiv.2402.00838 (accessed on 24 April 2026)
OLMoE: Open Mixture-of-Experts Language Models	https://doi.org/10.48550/arXiv.2409.02060 (accessed on 24 April 2026)
OpenELM: An Efficient Language Model Family with Open Training and Inference Framework	https://doi.org/10.48550/arXiv.2404.14619 (accessed on 24 April 2026)
Pangu Embedded: An Efficient Dual-system LLM Reasoner with Metacognition	https://doi.org/10.48550/arXiv.2505.22375 (accessed on 24 April 2026)
Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone	https://doi.org/10.48550/arXiv.2404.14219 (accessed on 24 April 2026)
Phi-3.5-MoE Technical Report	https://doi.org/10.48550/arXiv.2404.14219 (accessed on 24 April 2026)
PLaMo 2 Technical Report	https://doi.org/10.48550/arXiv.2509.04897 (accessed on 24 April 2026)
PLM: Efficient Peripheral Language Models Hardware-Co-Designed for Ubiquitous Computing	https://doi.org/10.48550/arXiv.2503.12167 (accessed on 24 April 2026)
Qwen Technical Report	https://doi.org/10.48550/arXiv.2309.16609 (accessed on 24 April 2026)
Qwen2 Technical Report	https://doi.org/10.48550/arXiv.2407.10671 (accessed on 24 April 2026)
Qwen3 Technical Report	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
Qwen3.5 MoE Transformers documentation (`qwen35moe`)	https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026)
Qwen3Guard Technical Report	https://doi.org/10.48550/arXiv.2510.14276 (accessed on 24 April 2026)
RADLADS: Rapid Attention Distillation to Linear Attention Decoders at Scale	https://doi.org/10.48550/arXiv.2505.03005 (accessed on 24 April 2026)
RWKV-7 "Goose" with Expressive Dynamic State Evolution	https://doi.org/10.48550/arXiv.2503.14456 (accessed on 24 April 2026)
SmallThinker: A Family of Efficient Large Language Models Natively Trained for Local Deployment	https://doi.org/10.48550/arXiv.2507.20984 (accessed on 24 April 2026)
Stable LM 2 1.6B Technical Report	https://doi.org/10.48550/arXiv.2402.17834 (accessed on 24 April 2026)

Table A2. *Cont.*

Resource	URL or DOI
StarCoder 2 and The Stack v2: The Next Generation	https://doi.org/10.48550/arXiv.2402.19173 (accessed on 24 April 2026)
StarCoder: may the source be with you!	https://doi.org/10.48550/arXiv.2305.06161 (accessed on 24 April 2026)
The Falcon Series of Open Language Models	https://doi.org/10.48550/arXiv.2311.16867 (accessed on 24 April 2026)

Table A3. Instruct model-resource links.

Resource	URL or DOI
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [26]	https://doi.org/10.18653/v1/N19-1423 (accessed on 24 April 2026)
The Llama 3 Herd of Models [28]	https://doi.org/10.48550/arXiv.2407.21783 (accessed on 24 April 2026)
Qwen3 Technical Report [30]	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
GPT-2 [33]	https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 24 April 2026)
DeciLM 6B [37]	https://huggingface.co/Deci/DeciLM-6b (accessed on 24 April 2026)
Jais 2 [41]	https://www.cerebras.ai/blog/jais2 (accessed on 24 April 2026)
MosaicBERT [44]	https://openreview.net/forum?id=5zipcfLC2Z (accessed on 24 April 2026)
Persimmon-8B [45]	https://www.adept.ai/blog/persimmon-8b/ (accessed on 24 April 2026)
Phi-2 [46]	https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ (accessed on 24 April 2026)
Apertus: Democratizing Open and Compliant LLMs for Global Language Environments	https://doi.org/10.48550/arXiv.2509.14233 (accessed on 24 April 2026)
Baichuan 2: Open Large-scale Language Models	https://doi.org/10.48550/arXiv.2309.10305 (accessed on 24 April 2026)
BLOOM: A 176B-Parameter Open-Access Multilingual Language Model	https://doi.org/10.48550/arXiv.2211.05100 (accessed on 24 April 2026)
diffuse-cpp LLaDA GGUF documentation (`diffuse`)	https://huggingface.co/diffuse-cpp/LLaDA-8B-Instruct-GGUF (accessed on 24 April 2026)
Dream 7B: Diffusion Large Language Models	https://doi.org/10.48550/arXiv.2508.15487 (accessed on 24 April 2026)
EuroBERT: Scaling Multilingual Encoders for European Languages	https://doi.org/10.48550/arXiv.2503.05500 (accessed on 24 April 2026)

Table A3. *Cont.*

Resource	URL or DOI
EXAONE 3.5: Series of Large Language Models for Real-world Use Cases	https://doi.org/10.48550/arXiv.2412.04862 (accessed on 24 April 2026)
Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance	https://doi.org/10.48550/arXiv.2507.22448 (accessed on 24 April 2026)
Gemma 2: Improving Open Language Models at a Practical Size	https://doi.org/10.48550/arXiv.2408.00118 (accessed on 24 April 2026)
Gemma: Open Models Based on Gemini Research and Technology	https://doi.org/10.48550/arXiv.2403.08295 (accessed on 24 April 2026)
GPT-NeoX-20B: An Open-Source Autoregressive Language Model	https://doi.org/10.18653/v1/2022.bigscience-1.9 (accessed on 24 April 2026)
Llama 2: Open Foundation and Fine-Tuned Chat Models	https://doi.org/10.48550/arXiv.2307.09288 (accessed on 24 April 2026)
Lumina-Image 2.0: A Unified and Efficient Image Generative Framework (`lumina2`)	https://doi.org/10.48550/arXiv.2503.21758 (accessed on 24 April 2026)
Ministral 3	https://doi.org/10.48550/arXiv.2601.08584 (accessed on 24 April 2026)
OLMo: Accelerating the Science of Language Models	https://doi.org/10.48550/arXiv.2402.00838 (accessed on 24 April 2026)
OpenELM: An Efficient Language Model Family with Open Training and Inference Framework	https://doi.org/10.48550/arXiv.2404.14619 (accessed on 24 April 2026)
Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone	https://doi.org/10.48550/arXiv.2404.14219 (accessed on 24 April 2026)
Qwen2 Technical Report	https://doi.org/10.48550/arXiv.2407.10671 (accessed on 24 April 2026)
StarCoder 2 and The Stack v2: The Next Generation	https://doi.org/10.48550/arXiv.2402.19173 (accessed on 24 April 2026)
StarCoder: may the source be with you!	https://doi.org/10.48550/arXiv.2305.06161 (accessed on 24 April 2026)
The Falcon Series of Open Language Models	https://doi.org/10.48550/arXiv.2311.16867 (accessed on 24 April 2026)

Table A4. Thinking model-resource links.

Resource	URL or DOI
Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution [13]	https://doi.org/10.48550/arXiv.2409.12191 (accessed on 24 April 2026)

Table A4. *Cont.*

Resource	URL or DOI
Granite 3.0 [25]	https://github.com/ibm-granite/granite-3.0-language-models/blob/main/paper.pdf (accessed on 24 April 2026)
Granite 3.0 MoE [25]	https://github.com/ibm-granite/granite-3.0-language-models/blob/main/paper.pdf (accessed on 24 April 2026)
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [26]	https://doi.org/10.18653/v1/N19-1423 (accessed on 24 April 2026)
Mamba [27]	https://openreview.net/forum?id=tEYskw1VY2 (accessed on 24 April 2026)
The Llama 3 Herd of Models [28]	https://doi.org/10.48550/arXiv.2407.21783 (accessed on 24 April 2026)
Qwen3 Technical Report [30]	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
Qwen3-VL Technical Report [32]	https://doi.org/10.48550/arXiv.2511.21631 (accessed on 24 April 2026)
GPT-2 [33]	https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 24 April 2026)
T5 [34]	https://www.jmlr.org/papers/v21/20-074.html (accessed on 24 April 2026)
AFM-4.5B [35]	https://www.arcee.ai/blog/deep-dive-afm-4-5b-the-first-arcee-foundational-model (accessed on 24 April 2026)
ERNIE 4.5 [38]	https://yiyan.baidu.com/blog/publication/ERNIE_Technical_Report.pdf (accessed on 24 April 2026)
Gemma 3n [39]	https://ai.google.dev/gemma/docs/gemma-3n (accessed on 24 April 2026)
Granite 4.0 Hybrid [40]	https://www.ibm.com/new/announcements/ibm-granite-4-0-hyper-efficient-high-performance-hybrid-models (accessed on 24 April 2026)
Jais 2 [41]	https://www.cerebras.ai/blog/jais2 (accessed on 24 April 2026)
MosaicBERT [44]	https://openreview.net/forum?id=5zipcfLC2Z (accessed on 24 April 2026)
Persimmon-8B [45]	https://www.adept.ai/blog/persimmon-8b/ (accessed on 24 April 2026)
Phi-2 [46]	https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ (accessed on 24 April 2026)
Qwen3.5 [48]	https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026)
SmolLM3 [50]	https://huggingface.co/blog/smollm3 (accessed on 24 April 2026)

Table A4. *Cont.*

Resource	URL or DOI
Baichuan 2: Open Large-scale Language Models	https://doi.org/10.48550/arXiv.2309.10305 (accessed on 24 April 2026)
BLOOM: A 176B-Parameter Open-Access Multilingual Language Model	https://doi.org/10.48550/arXiv.2211.05100 (accessed on 24 April 2026)
Chameleon: Mixed-Modal Early-Fusion Foundation Models	https://doi.org/10.48550/arXiv.2405.09818 (accessed on 24 April 2026)
ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools	https://doi.org/10.48550/arXiv.2406.12793 (accessed on 24 April 2026)
EuroBERT: Scaling Multilingual Encoders for European Languages	https://doi.org/10.48550/arXiv.2503.05500 (accessed on 24 April 2026)
EXAONE 3.5: Series of Large Language Models for Real-world Use Cases	https://doi.org/10.48550/arXiv.2412.04862 (accessed on 24 April 2026)
Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance	https://doi.org/10.48550/arXiv.2507.22448 (accessed on 24 April 2026)
Gemma 2: Improving Open Language Models at a Practical Size	https://doi.org/10.48550/arXiv.2408.00118 (accessed on 24 April 2026)
Gemma 3 Technical Report	https://doi.org/10.48550/arXiv.2503.19786 (accessed on 24 April 2026)
Gemma 4 model card (`gemma4`)	https://huggingface.co/google/gemma-4-E2B-it (accessed on 24 April 2026)
Gemma: Open Models Based on Gemini Research and Technology	https://doi.org/10.48550/arXiv.2403.08295 (accessed on 24 April 2026)
GPT-NeoX-20B: An Open-Source Autoregressive Language Model	https://doi.org/10.18653/v1/2022.bigscience-1.9 (accessed on 24 April 2026)
gpt-oss-120b/gpt-oss-20b Model Card	https://doi.org/10.48550/arXiv.2508.10925 (accessed on 24 April 2026)
InternLM2 Technical Report	https://doi.org/10.48550/arXiv.2403.17297 (accessed on 24 April 2026)
Jamba: A Hybrid Transformer-Mamba Language Model	https://doi.org/10.48550/arXiv.2403.19887 (accessed on 24 April 2026)
LFM2 Technical Report	https://doi.org/10.48550/arXiv.2511.23404 (accessed on 24 April 2026)
Llama 2: Open Foundation and Fine-Tuned Chat Models	https://doi.org/10.48550/arXiv.2307.09288 (accessed on 24 April 2026)
Llama-Embed-Nemotron-8B: A Universal Text Embedding Model for Multilingual and Cross-Lingual Tasks	https://doi.org/10.48550/arXiv.2511.07025 (accessed on 24 April 2026)
Ministral 3	https://doi.org/10.48550/arXiv.2601.08584 (accessed on 24 April 2026)

Table A4. *Cont.*

Resource	URL or DOI
Nemotron-H: A Family of Accurate and Efficient Hybrid Mamba-Transformer Models	https://doi.org/10.48550/arXiv.2504.03624 (accessed on 24 April 2026)
OLMo 2: The Best Fully Open Language Model to Date	https://doi.org/10.48550/arXiv.2501.00656 (accessed on 24 April 2026)
Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone	https://doi.org/10.48550/arXiv.2404.14219 (accessed on 24 April 2026)
Qwen2 Technical Report	https://doi.org/10.48550/arXiv.2407.10671 (accessed on 24 April 2026)
Qwen3 Technical Report	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference	https://doi.org/10.18653/v1/2025.acl-long.127 (accessed on 24 April 2026)
Stable LM 2 1.6B Technical Report	https://doi.org/10.48550/arXiv.2402.17834 (accessed on 24 April 2026)
StarCoder 2 and The Stack v2: The Next Generation	https://doi.org/10.48550/arXiv.2402.19173 (accessed on 24 April 2026)

Table A5. Audio model-resource links.

Resource	URL or DOI
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [26]	https://doi.org/10.18653/v1/N19-1423 (accessed on 24 April 2026)
The Llama 3 Herd of Models [28]	https://doi.org/10.48550/arXiv.2407.21783 (accessed on 24 April 2026)
WavTokenizer [29]	https://openreview.net/forum?id=yBlVlS2Fd9 (accessed on 24 April 2026)
Qwen3 Technical Report [30]	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
Qwen3-VL Technical Report [32]	https://doi.org/10.48550/arXiv.2511.21631 (accessed on 24 April 2026)
GPT-2 [33]	https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 24 April 2026)
Gemma 3n [39]	https://ai.google.dev/gemma/docs/gemma-3n (accessed on 24 April 2026)
Phi-2 [46]	https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ (accessed on 24 April 2026)
Qwen3.5 [48]	https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026)

Table A5. *Cont.*

Resource	URL or DOI
Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance	https://doi.org/10.48550/arXiv.2507.22448 (accessed on 24 April 2026)
Fish-Speech: Leveraging Large Language Models for Advanced Multilingual Text-to-Speech Synthesis	https://doi.org/10.48550/arXiv.2411.01156 (accessed on 24 April 2026)
Gemma 2: Improving Open Language Models at a Practical Size	https://doi.org/10.48550/arXiv.2408.00118 (accessed on 24 April 2026)
Gemma 3 Technical Report	https://doi.org/10.48550/arXiv.2503.19786 (accessed on 24 April 2026)
Gemma 4 model card (`gemma4`)	https://huggingface.co/google/gemma-4-E2B-it (accessed on 24 April 2026)
Gemma: Open Models Based on Gemini Research and Technology	https://doi.org/10.48550/arXiv.2403.08295 (accessed on 24 April 2026)
GPT-NeoX-20B: An Open-Source Autoregressive Language Model	https://doi.org/10.18653/v1/2022.bigscience-1.9 (accessed on 24 April 2026)
jina-embeddings-v3: Multilingual Embeddings With Task LoRA	https://doi.org/10.48550/arXiv.2409.10173 (accessed on 24 April 2026)
LFM2 Technical Report	https://doi.org/10.48550/arXiv.2511.23404 (accessed on 24 April 2026)
Llama 2: Open Foundation and Fine-Tuned Chat Models	https://doi.org/10.48550/arXiv.2307.09288 (accessed on 24 April 2026)
MOSS-TTS Technical Report (`moss-tts-delay`)	https://doi.org/10.48550/arXiv.2603.18090 (accessed on 24 April 2026)
OLMo: Accelerating the Science of Language Models	https://doi.org/10.48550/arXiv.2402.00838 (accessed on 24 April 2026)
Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone	https://doi.org/10.48550/arXiv.2404.14219 (accessed on 24 April 2026)
pig GGUF encoder documentation (`pig`)	https://huggingface.co/calcuis/pig-encoder (accessed on 24 April 2026)
Qwen2 Technical Report	https://doi.org/10.48550/arXiv.2407.10671 (accessed on 24 April 2026)
Qwen3-ASR Technical Report	https://doi.org/10.48550/arXiv.2601.21337 (accessed on 24 April 2026)
Qwen3-TTS Technical Report	https://doi.org/10.48550/arXiv.2601.15621 (accessed on 24 April 2026)
Voxtral Realtime Transformers documentation (`voxtral_realtime`)	https://doi.org/10.48550/arXiv.2602.11298 (accessed on 24 April 2026)
XTTS: a Massively Multilingual Zero-Shot Text-to-Speech Model	https://doi.org/10.48550/arXiv.2406.04904 (accessed on 24 April 2026)

Table A6. Vision-language model-resource links.

Resource	URL or DOI
Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution [13]	https://doi.org/10.48550/arXiv.2409.12191 (accessed on 24 April 2026)
Granite 3.0 [25]	https://github.com/ibm-granite/granite-3.0-language-models/blob/main/paper.pdf (accessed on 24 April 2026)
The Llama 3 Herd of Models [28]	https://doi.org/10.48550/arXiv.2407.21783 (accessed on 24 April 2026)
Qwen3 Technical Report [30]	https://doi.org/10.48550/arXiv.2505.09388 (accessed on 24 April 2026)
Qwen3-VL Technical Report [32]	https://doi.org/10.48550/arXiv.2511.21631 (accessed on 24 April 2026)
GPT-2 [33]	https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 24 April 2026)
Gemma 3n [39]	https://ai.google.dev/gemma/docs/gemma-3n (accessed on 24 April 2026)
Phi-2 [46]	https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ (accessed on 24 April 2026)
Qwen3.5 [48]	https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026)
VAETKI [52]	https://huggingface.co/NC-AI-consortium-VAETKI/VAETKI (accessed on 24 April 2026)
ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools	https://doi.org/10.48550/arXiv.2406.12793 (accessed on 24 April 2026)
DeepSeek-OCR: Contexts Optical Compression	https://doi.org/10.48550/arXiv.2601.20552 (accessed on 24 April 2026)
DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model	https://doi.org/10.48550/arXiv.2405.04434 (accessed on 24 April 2026)
Gemma 3 Technical Report	https://doi.org/10.48550/arXiv.2503.19786 (accessed on 24 April 2026)
Gemma 4 model card (`gemma4`)	https://huggingface.co/google/gemma-4-E2B-it (accessed on 24 April 2026)
Gemma: Open Models Based on Gemini Research and Technology	https://doi.org/10.48550/arXiv.2403.08295 (accessed on 24 April 2026)
Hunyuan-Large: An Open-Source MoE Model with 52 Billion Activated Parameters	https://doi.org/10.48550/arXiv.2411.02265 (accessed on 24 April 2026)
LFM2 Technical Report	https://doi.org/10.48550/arXiv.2511.23404 (accessed on 24 April 2026)

Table A6. *Cont.*

Resource	URL or DOI
Llama 2: Open Foundation and Fine-Tuned Chat Models	https://doi.org/10.48550/arXiv.2307.09288 (accessed on 24 April 2026)
MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies	https://doi.org/10.48550/arXiv.2404.06395 (accessed on 24 April 2026)
Ministral 3	https://doi.org/10.48550/arXiv.2601.08584 (accessed on 24 April 2026)
PaddleOCR 3.0 Technical Report	https://doi.org/10.48550/arXiv.2507.05595 (accessed on 24 April 2026)
Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone	https://doi.org/10.48550/arXiv.2404.14219 (accessed on 24 April 2026)
pig GGUF encoder documentation (`pig`)	https://huggingface.co/calcuis/pig-encoder (accessed on 24 April 2026)
Qwen2 Technical Report	https://doi.org/10.48550/arXiv.2407.10671 (accessed on 24 April 2026)
Qwen3.5 MoE Transformers documentation (`qwen35moe`)	https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026)
RWKV-7 "Goose" with Expressive Dynamic State Evolution	https://doi.org/10.48550/arXiv.2503.14456 (accessed on 24 April 2026)
SeedVR2: One-Step Video Restoration via Diffusion Adversarial Post-Training (`seedvr`)	https://doi.org/10.48550/arXiv.2506.05301 (accessed on 24 April 2026)

Table A7. Vision-language-action model-resource links.

Resource	URL or DOI
Helix VLA [7]	https://www.figure.ai/news/helix (accessed on 24 April 2026)
OpenVLA [8]	https://proceedings.mlr.press/v270/kim25c.html (accessed on 24 April 2026)
SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics [9]	https://doi.org/10.48550/arXiv.2506.01844 (accessed on 24 April 2026)
Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution [13]	https://doi.org/10.48550/arXiv.2409.12191 (accessed on 24 April 2026)
The Llama 3 Herd of Models [28]	https://doi.org/10.48550/arXiv.2407.21783 (accessed on 24 April 2026)
Qwen3-VL Technical Report [32]	https://doi.org/10.48550/arXiv.2511.21631 (accessed on 24 April 2026)
Llama 2: Open Foundation and Fine-Tuned Chat Models	https://doi.org/10.48550/arXiv.2307.09288 (accessed on 24 April 2026)

References

1. Zheng, Y.; Chen, Y.; Qian, B.; Shi, X.; Shu, Y.; Chen, J. A Review on Edge Large Language Models: Design, Execution, and Applications. *ACM Comput. Surv.* **2025**, *57*, 209. [CrossRef]
2. Wang, R.; Gao, Z.; Zhang, L.; Yue, S.; Gao, Z. Empowering Large Language Models to Edge Intelligence: A Survey of Edge Efficient LLMs and Techniques. *Comput. Sci. Rev.* **2025**, *57*, 100755. [CrossRef]
3. Lee, Y.; Golden, A.; Sun, A.; Hosmer, B.; Acun, B.; Balioglu, C.; Wang, C.; Hernandez, C.D.; Puhrsch, C.; Haziza, D.; et al. Characterizing and Efficiently Accelerating Multimodal Generation Model Inference. *IEEE Micro* **2025**, *45*, 103–115. [CrossRef]
4. Dhar, N.; Deng, B.; Lo, D.; Wu, X.; Zhao, L.; Suo, K. An Empirical Analysis and Resource Footprint Study of Deploying Large Language Models on Edge Devices. In *Proceedings of the 2024 ACM Southeast Conference*; ACM: New York, NY, USA, 2024; pp. 69–76. [CrossRef]
5. Laskaridis, S.; Katevas, K.; Minto, L.; Haddadi, H. MELTing Point: Mobile Evaluation of Language Transformers. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*; ACM: New York, NY, USA, 2024; pp. 890–907. [CrossRef]
6. Ribar, L.; Chelombiev, I.; Hudlass-Galley, L.; Blake, C.; Luschi, C.; Orr, D. SparQ Attention: Bandwidth-Efficient LLM Inference. In *Proceedings of the 41st International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2024; Volume 235, pp. 42558–42583. Available online: https://proceedings.mlr.press/v235/ribar24a.html (accessed on 24 April 2026).
7. Figure AI. Helix: A Vision-Language-Action Model for Generalist Humanoid Control. Company Webpage. 2025. Available online: https://www.figure.ai/news/helix (accessed on 24 April 2026).
8. Kim, M.J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proceedings of the 8th Conference on Robot Learning*; PMLR: Cambridge, MA, USA, 2025; Volume 270, pp. 2679–2713. Available online: https://proceedings.mlr.press/v270/kim25c.html (accessed on 24 April 2026).
9. Shukor, M.; Aubakirova, D.; Capuano, F.; Kooijmans, P.; Palma, S.; Zouitine, A.; Aractingi, M.; Pascal, C.; Russi, M.; Marafioti, A.; et al. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. *arXiv* **2025**, arXiv:2506.01844. [CrossRef]
10. Hugging Face Hub. GGUF. Project Documentation. 2026. Available online: https://huggingface.co/docs/hub/gguf (accessed on 24 April 2026).
11. ggml-org. llama.cpp. Project Repository. 2026. Available online: https://github.com/ggml-org/llama.cpp (accessed on 24 April 2026).
12. GPUStack. gguf-parser-go. Project Repository. 2026. Available online: https://github.com/gpustack/gguf-parser-go (accessed on 24 April 2026).
13. Wang, P.; Bai, S.; Tan, S.; Wang, S.; Fan, Z.; Bai, J.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; et al. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv* **2024**, arXiv:2409.12191. [CrossRef]
14. Cai, G.; Tian, R.; Yang, L.; Jia, Y.; Li, L.; Wang, J. Efficient Inference for Edge Large Language Models: A Survey. *Tsinghua Sci. Technol.* **2026**, *31*, 1365–1380. [CrossRef]
15. Hugging Face. hf-agents. Project Repository. 2026. Available online: https://github.com/huggingface/hf-agents/tree/main (accessed on 24 April 2026).
16. ggml-org. GGUF File Format Specification. Project Documentation. 2026. Available online: https://github.com/ggml-org/ggml/blob/master/docs/gguf.md (accessed on 24 April 2026).
17. Welch, B. On the Comparison of Several Mean Values: An Alternative Approach. *Biometrika* **1951**, *38*, 330–336. [CrossRef]
18. Statsmodels Developers. statsmodels.stats.oneway.anova_oneway. Software Documentation. 2026. Available online: https://www.statsmodels.org/dev/generated/statsmodels.stats.oneway.anova_oneway.html (accessed on 24 April 2026).
19. Games, P.A.; Howell, J.F. Pairwise Multiple Comparison Procedures with Unequal N's and/or Variances: A Monte Carlo Study. *J. Educ. Stat.* **1976**, *1*, 113–125. [CrossRef]
20. Kruskal, W.H.; Wallis, W.A. Use of Ranks in One-Criterion Variance Analysis. *J. Am. Stat. Assoc.* **1952**, *47*, 583–621. [CrossRef]
21. Dunn, O. Multiple Comparisons Using Rank Sums. *Technometrics* **1964**, *6*, 241–252. [CrossRef]
22. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
23. Tomczak, M.; Tomczak, E. The Need to Report Effect Size Estimates Revisited: An Overview of Some Recommended Measures of Effect Size. *Trends Sport Sci.* **2014**, *21*, 19–25.
24. Cliff, N. Dominance Statistics: Ordinal Analyses to Answer Ordinal Questions. *Psychol. Bull.* **1993**, *114*, 494–509. [CrossRef]
25. Granite Team. Granite 3.0 Language Models. 2024. Available online: https://github.com/ibm-granite/granite-3.0-language-models/blob/main/paper.pdf (accessed on 24 April 2026).
26. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; Volume 1; pp. 4171–4186. [CrossRef]

27. Gu, A.; Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. Conference on Language Modeling (COLM), OpenReview. 2024. Available online: https://openreview.net/forum?id=tEYskw1VY2 (accessed on 24 April 2026).
28. Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. The Llama 3 Herd of Models. *arXiv* **2024**, arXiv:2407.21783. [CrossRef]
29. Ji, S.; Jiang, Z.; Wang, W.; Chen, Y.; Fang, M.; Zuo, J.; Yang, Q.; Cheng, X.; Wang, Z.; Li, R.; et al. WavTokenizer: An Efficient Acoustic Discrete Codec Tokenizer for Audio Language Modeling. In *International Conference on Learning Representations (ICLR)*; OpenReview: Amherst, MA, USA, 2025. Available online: https://openreview.net/forum?id=yBlVlS2Fd9 (accessed on 24 April 2026).
30. Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. Qwen3 Technical Report. *arXiv* **2025**, arXiv:2505.09388. [CrossRef]
31. Zuo, J.; Velikanov, M.; Chahed, I.; Belkada, Y.; Rhayem, D.E.; Kunsch, G.; Hacid, H.; Yous, H.; Farhat, B.; Khadraoui, I.; et al. Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance. *arXiv* **2025**, arXiv:2507.22448. [CrossRef]
32. Bai, S.; Cai, Y.; Chen, R.; Chen, K.; Chen, X.; Cheng, Z.; Deng, L.; Ding, W.; Gao, C.; Ge, C.; et al. Qwen3-VL Technical Report. *arXiv* **2025**, arXiv:2511.21631. [CrossRef]
33. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. Technical Report, OpenAI. 2019. Available online: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 24 April 2026).
34. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67. Available online: https://www.jmlr.org/papers/v21/20-074.html (accessed on 24 April 2026).
35. Arcee AI. AFM-4.5B. Model Card. 2025. Available online: https://www.arcee.ai/blog/deep-dive-afm-4-5b-the-first-arcee-foundational-model (accessed on 24 April 2026).
36. Cohere. Command R: Retrieval-Augmented Generation at Production Scale. Product/Release Note. 2024. Available online: https://huggingface.co/CohereLabs/c4ai-command-r-v01 (accessed on 24 April 2026).
37. Deci AI. DeciLM 6B. Model Card, Deci. 2023. Available online: https://huggingface.co/Deci/DeciLM-6b (accessed on 24 April 2026).
38. Baidu ERNIE Team. ERNIE 4.5 Technical Report. Technical Release, Baidu. 2025. Available online: https://yiyan.baidu.com/blog/publication/ERNIE_Technical_Report.pdf (accessed on 24 April 2026).
39. Gemma Team. Gemma 3n: Efficient On-Device Models. Developer/Model Overview, Google AI for Developers. 2025. Available online: https://ai.google.dev/gemma/docs/gemma-3n (accessed on 24 April 2026).
40. IBM Granite Team. IBM Granite 4.0 Hybrid Language Models. Model Announcement, IBM. 2025. Available online: https://www.ibm.com/new/announcements/ibm-granite-4-0-hyper-efficient-high-performance-hybrid-models (accessed on 24 April 2026).
41. Inception/G42. Jais 2: A Blueprint for Sovereign AI. Release Page, Cerebras/Inception/G42/MBZUAI. 2025. Available online: https://www.cerebras.ai/blog/jais2 (accessed on 24 April 2026).
42. Meta AI. The Llama 4 Herd of Models. Model Announcement, Meta. 2025. Available online: https://ai.meta.com/blog/llama-4-multimodal-intelligence/ (accessed on 24 April 2026).
43. Maincode Team. Maincoder-1B: A High-Performance 1B Parameter Coding Model. Model Card, Maincode. 2025. Available online: https://huggingface.co/Maincode/Maincoder-1B (accessed on 24 April 2026).
44. Portes, J.; Trott, A.; Havens, S.; King, D.; Venigalla, A.; Nadeem, M.; Sardana, N.; Khudia, D.; Frankle, J. MosaicBERT: A Bidirectional Encoder Optimized for Fast Pretraining. 2023. Available online: https://openreview.net/forum?id=5zipcfLC2Z (accessed on 24 April 2026).
45. Adept AI. Releasing Persimmon-8B. Release Page, Adept. 2023. Available online: https://www.adept.ai/blog/persimmon-8b/ (accessed on 24 April 2026).
46. Microsoft Research. Phi-2: The Surprising Power of Small Language Models. Technical Blog, Microsoft. 2023. Available online: https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/ (accessed on 24 April 2026).
47. Preferred Networks, Inc. Next-Generation Large Language Model PLaMo 3 Series. Preferred Networks Tech Blog. Official Release Page for PLaMo 3 NICT 8B and 31B Models. 2025. Available online: https://tech.preferred.jp/ja/blog/plamo_3_8b_31b/ (accessed on 24 April 2026).
48. Qwen Team. Qwen3.5: Towards Native Multimodal Agents. 2026. Available online: https://qwen.ai/blog?id=qwen3.5 (accessed on 24 April 2026).
49. Essential AI. Rnj-1: Building Instruments of Intelligence. Research Blog Post. 2025. Available online: https://essential.ai/research/rnj-1 (accessed on 24 April 2026).

50. Hugging Face. SmolLM3: Smol, Multilingual, Long-Context Reasoner. Release Blog. 2025. Available online: https://huggingface.co/blog/smollm3 (accessed on 24 April 2026).
51. XVERSE AI. XVERSE-7B. Model Card. 2023. Available online: https://huggingface.co/xverse/XVERSE-7B (accessed on 24 April 2026).
52. NC-AI Consortium. VAETKI. Model Card. 2025. Available online: https://huggingface.co/NC-AI-consortium-VAETKI/VAETKI (accessed on 24 April 2026).