International Journal of Artificial Intelligence
p-ISSN: 2407-7275, e-ISSN: 2686-3251

Original Research Paper

Capability-Specific Degradation Patterns in Quantized Small Language Models

Emil Rahimov[1*]

[1] *Department of AI and Data Science, Digital Research Lab. Nakhchivan, Azerbaijan.*

Article History
Received:
09.02.2026

Revised:
13.03.2026

Accepted:
29.03.2026

***Corresponding Author:**
Emil Rahimov
Email
emilrahimov25@gmail.com

Abstract: Post-training quantization is the default method for deploying small language models (one to four billion parameters) on consumer and edge hardware, yet its effect is usually summarized by a single aggregate accuracy score that can conceal severe failures in individual capabilities. This paper presents a capability-level analysis of four-bit quantization for seven open instruction-tuned small language models drawn from five architecture families: Qwen2.5, Llama-3.2, Gemma-2, Phi-3.5, and SmolLM2. Each model is evaluated at sixteen-bit floating point and at four-bit precision across six capabilities, namely factual knowledge, commonsense reasoning, mathematical reasoning, multilingual mathematical reasoning, code generation, and instruction following, yielding eighty-four controlled evaluations. Instead of reporting only mean accuracy, we construct a per-capability degradation map and test, using Kendall's rank correlation, whether capabilities degrade in a consistent order across architectures. The results show that degradation is strongly capability-specific: multilingual mathematical reasoning and code generation are the most fragile capabilities, with relative losses of up to fifty-seven percent, whereas commonsense reasoning is almost entirely preserved. However, the ordering of degradation is only weakly consistent across architectures, with a mean Kendall's tau of 0.29, indicating that the most and least fragile capabilities are shared but the overall ranking is architecture-dependent. Smaller models degrade more in magnitude. We conclude that small-model quantization should be evaluated per capability rather than in aggregate.

Keywords: Code Generation, Edge Artificial Intelligence, Model Quantization, Multilingual Reasoning, Small Language Models.

1. Introduction

Transformer-based large language models (LLMs) [1] have progressed rapidly from research artifacts into deployed components of everyday software [2] [3]. As capable open models in the one-to-four-billion-parameter range become available, a growing share of practical deployment occurs on consumer graphics processors, laptops, and edge devices, where memory and latency budgets are tight. Post-training quantization is the standard tool for meeting these budgets: by representing model weights in eight or four bits rather than sixteen, quantization reduces memory footprint and bandwidth with little or no retraining [4] [5] [6] [7]. For a practitioner, the operative question is therefore not whether to quantize a small model, but what is lost when it is quantized.

The community usually answers this question with an aggregate metric: a single accuracy number, or an average over a benchmark suite. We argue that this granularity is misleading for deployment. A model whose mean accuracy falls by a few points may have lost almost nothing on one capability while losing a large fraction of another. Recent evaluation-focused studies support this concern: quantization affects tasks unevenly [8] [9] and disproportionately harms multilingual generation [10], while empirical studies of specific model families report degradation concentrated in particular abilities [11]. Existing analyses, however, tend to compare quantization methods, to focus on a single capability or family, or to target larger models, leaving the deployment-relevant small-model regime and the question of cross-architecture consistency underexplored.

This paper studies four-bit quantization at the level of individual capabilities for small instruction-tuned LLMs. We evaluate seven models from five architecture families at sixteen-bit and four-bit precision across six capabilities, and organize the outcome as a per-capability degradation map. We then ask whether capabilities degrade in a consistent order across architectures, using Kendall's rank correlation. The contributions are: (1) a controlled capability-level degradation map for seven small LLMs under four-bit quantization; (2) evidence that degradation is strongly capability-specific, with multilingual mathematical reasoning and code generation most fragile and commonsense reasoning most robust; (3) a cross-architecture consistency analysis showing the degradation ordering is only weakly universal, with a mean Kendall's tau of 0.29; and (4) a within-family size analysis and a demonstration that aggregate scores conceal capability-specific collapse, including a prompting-protocol pitfall in code evaluation.

2. Literature Review

Quantization for LLMs spans weight-only and weight-activation methods. GPTQ [5] performs accurate layer-wise weight quantization, AWQ [6] preserves salient weight channels, and the bitsandbytes NF4 scheme [7] enables four-bit storage with on-the-fly dequantization. Weight-activation methods such as SmoothQuant [12] and ZeroQuant [13] address activation outliers, scaling-law analyses [14] characterize the accuracy-versus-bit trade-off, and rotation- or sparsity-based methods such as QuaRot [15], SqueezeLLM [16], and SparseGPT [17] push the low-bit frontier. These build on a longer line of neural-network compression research, including integer-arithmetic-only inference [18], deep compression [19], quantized neural networks [20], low-bit post-training quantization of convolutional networks [21], adaptive rounding [22], and transformer-specific quantization for BERT-style models [23] [24]; recent surveys consolidate compression techniques for LLMs [25], and parameter-efficient adaptation such as LoRA [26] is frequently combined with quantization for memory-constrained fine-tuning [7]. Complementary to method development, a growing body of work evaluates the downstream impact of quantization through cross-strategy comparisons [8] [9], multilingual analyses [10], and family-specific empirical studies [11]. Our work belongs to this evaluation tradition but is distinguished by an exclusive focus on the small (at or below four billion parameters) regime, five architecture families under one identical pipeline, an explicit capability degradation ordering with a test of its cross-architecture consistency, and a within-family size axis.

3. Methodology

3.1. Models

We evaluate seven instruction-tuned models, all at or below four billion parameters. Five constitute the cross-architecture set used for the consistency analysis: Qwen2.5-3B-Instruct [27], Llama-3.2-3B-Instruct [28], Gemma-2-2B-it [29], Phi-3.5-mini-instruct (3.8 billion) [30], and SmolLM2-1.7B-Instruct [31]. Two further models, Qwen2.5-1.5B-Instruct [27] and Llama-3.2-1B-Instruct [28], form

a within-family size axis. Instruction-tuned variants are used because instruction following is only meaningful for them and because they are the deployment-relevant artifact at this scale. Gemma-2 is loaded with eager attention so that its logit soft-capping is applied correctly.

3.2. Quantization

We compare two precisions: sixteen-bit floating point (FP16), used as the reference, and four-bit weight quantization using the bitsandbytes NF4 scheme [7]. We focus on the four-bit regime because it is where capability-specific failures emerge; eight-bit quantization was near-lossless in preliminary runs and is omitted. The quantization method is held fixed across all architectures so that architecture is the only variable; GPTQ [5] and AWQ [6] are alternative four-bit methods that are not varied here.

3.3. Capabilities and Benchmarks

Six capabilities are each mapped to a standard benchmark and evaluated with the Language Model Evaluation Harness [32]: factual knowledge with ARC-Challenge [33] (25-shot, length-normalized accuracy); commonsense reasoning with HellaSwag [34] (10-shot, length-normalized accuracy); mathematical reasoning with GSM8K [35] (8-shot, exact match); multilingual mathematical reasoning with MGSM [36] (8-shot, exact match, macro-averaged over the standard languages); code generation with HumanEval [37] (zero-shot, pass@1); and instruction following with IFEval [38] (zero-shot, prompt-level strict accuracy). These six were chosen to span distinct abilities rather than the single aggregate scores such as MMLU [39] that are commonly reported; chain-of-thought-style reasoning [40] is implicit in the few-shot generative tasks. Generative tasks use the model's chat template; HumanEval uses raw completion following common practice; multiple-choice tasks use log-likelihood scoring.

3.4. Evaluation Pipeline

All runs use one controlled pipeline. Generative tasks are executed with the vLLM inference engine [41] for throughput; multiple-choice log-likelihood tasks use a standard Transformers backend. The same backend is used for a given capability across all precisions and models, so that any difference between FP16 and four-bit within a capability reflects quantization rather than the harness. Pipeline correctness was verified by matching FP16 baselines against published values. All seven models, two precisions, and six benchmarks (eighty-four evaluations) completed successfully.

3.5. Analysis

For each model and capability, we compute the relative degradation as the difference between the FP16 and four-bit scores divided by the FP16 score, expressed as a percentage. To avoid unstable ratios, any capability whose FP16 baseline is below ten points is excluded from the relative analysis and reported only as an absolute score. Capabilities are ranked by relative degradation within each model (rank one denotes the most degraded), and rankings are compared across model pairs using Kendall's rank correlation coefficient [42]. A high mean coefficient indicates a universal degradation order; a low mean indicates architecture dependence. We separately compute the coefficient between the small and large member of each family.

4. Finding and Discussion

4.1. Capability-specific degradation

Table 1 reports the relative degradation map and Table 2 the underlying absolute scores. The central observation is the very large spread within a single model. For Qwen2.5-1.5B, relative degradation ranges from 2.6 percent (HellaSwag) to 22.9 percent (MGSM) and 22.6 percent (HumanEval), an order-of-magnitude difference between the most and least affected capability. Collapsing these into one number would report a moderate drop and erase the fact that two capabilities lost roughly a quarter of their performance. Two capabilities stand out as consistently fragile: multilingual mathematical reasoning (MGSM) is the most degraded capability for five of the six models for which it is measurable, with relative drops of 12 to 23 percent, and code generation (HumanEval) is the most volatile, catastrophic for some models (Qwen2.5-1.5B 22.6 percent, Phi-3.5 21.0 percent, and SmolLM2 57 percent under the protocol discussed in Section 3.4) yet statistically unchanged for others (Llama-3.2-3B, Gemma-2). At the other extreme, commonsense reasoning (HellaSwag) is essentially preserved everywhere (1.7 to 5.1 percent), and factual knowledge (ARC) is largely robust.

Small negative values, where the four-bit score marginally exceeds FP16, occur for a few code and math cells and are within evaluation noise. Figure 1 visualizes the map.

Table 1 shows the elative degradation (%) from FP16 to four-bit (higher indicates worse). A dash marks capabilities excluded for an FP16 baseline below ten points. The dagger marks SmolLM2 HumanEval under the chat-template protocol.

Table 1. Relative Degradation (%) from FP16 to Four-Bit

Model	GSM8K	ARC	HellaSwag	HumanEval	IFEval	MGSM
Qwen2.5-3B	-2.6	3.4	2.6	9.2	1.9	19.0
Qwen2.5-1.5B	10.6	9.3	2.6	22.6	11.8	22.9
Llama-3.2-3B	4.0	3.1	1.8	-6.1	-0.5	19.1
Llama-3.2-1B	16.7	2.7	5.1	12.3	4.5	--
Gemma-2-2B	4.1	2.4	1.8	-2.6	1.0	12.2
Phi-3.5-mini	4.3	-1.7	1.7	21.0	1.1	19.0
SmolLM2-1.7B	18.1	2.7	2.4	57.2†	-1.2	--

While Table 2 shows the absolute scores (%) per model and precision. The dagger marks SmolLM2 HumanEval under the chat-template protocol; under raw completion both values are near zero.

Table 2. Absolute scores (%) per model and precision.

Model	GSM8K	ARC	HellaSwag	HumanEval	IFEval	MGSM
Qwen2.5-3B	FP16	63.4	60.7	75.3	53.1	59.3
Qwen2.5-3B	4-bit	65.1	58.6	73.3	48.2	58.2
Qwen2.5-1.5B	FP16	61.7	54.4	67.9	37.8	40.9
Qwen2.5-1.5B	4-bit	55.2	49.3	66.1	29.3	36.0
Llama-3.2-3B	FP16	77.3	52.8	73.6	50.0	69.3
Llama-3.2-3B	4-bit	74.2	51.2	72.3	53.1	69.7
Llama-3.2-1B	FP16	44.1	41.8	61.1	34.8	49.9
Llama-3.2-1B	4-bit	36.8	40.7	57.9	30.5	47.7
Gemma-2-2B	FP16	54.1	57.9	71.5	23.8	55.6
Gemma-2-2B	4-bit	51.9	56.6	70.2	24.4	55.1
Phi-3.5-mini	FP16	78.6	64.8	79.0	61.0	49.9
Phi-3.5-mini	4-bit	75.3	65.9	77.7	48.2	49.4
SmolLM2-1.7B	FP16	49.1	53.2	72.6	34.2†	48.2
SmolLM2-1.7B	4-bit	40.3	51.8	70.8	14.6†	48.8

Figure 1 shows the relative degradation (%) per model (rows) and capability (columns). Multilingual math and code are the hot columns; commonsense is uniformly cool.

4.2. Cross-Architecture Consistency

We rank the six capabilities by degradation within each model and compare rankings across all twenty-one model pairs using Kendall's rank correlation. The mean pairwise coefficient is 0.29, which we interpret as weak consistency: the degradation order is not universal across architectures. The pattern is better described as partially consistent. The extremes are shared, with multilingual mathematical reasoning reliably near the top of the ranking and commonsense reasoning reliably near the bottom, but the middle of the ranking (knowledge, math, code, and instruction following) reorders from architecture to architecture, which depresses the mean coefficient. Pairwise agreement is correspondingly heterogeneous: some pairs rank capabilities identically (for example, Gemma-2 and Llama-3.2-3B, with a coefficient of 1.0) while others are essentially uncorrelated. The practical implication is that one cannot reliably predict a new architecture's full degradation profile from another's; a deployment decision should measure the specific model rather than assume a universal ordering.

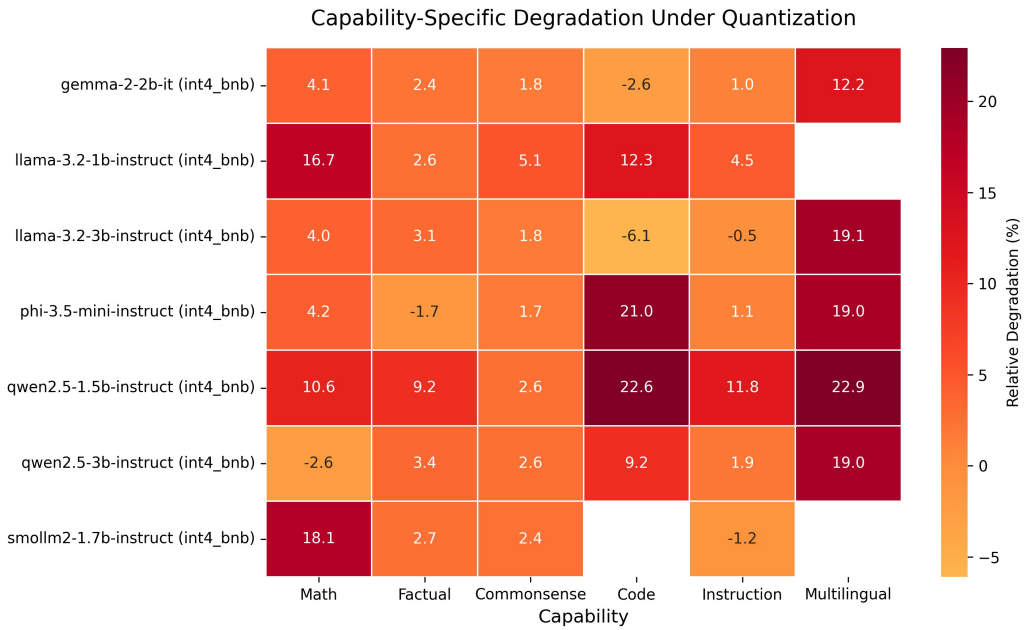

Figure 1. Relative Degradation Per Model and Capability

4.3. Effect of Model Size

Within the two families evaluated at two sizes, the smaller model degrades more in magnitude. Qwen2.5-1.5B degrades more than Qwen2.5-3B on five of six capabilities (for example, GSM8K 10.6 versus minus 2.6 percent, and HumanEval 22.6 versus 9.2 percent), and Llama-3.2-1B degrades more than Llama-3.2-3B on mathematics and code (GSM8K 16.7 versus 4.0 percent). The degradation ordering, however, does not strongly persist across sizes: the small-versus-large rank correlation is 0.47 for Qwen2.5 and 0.00 for Llama-3.2. Quantization therefore harms smaller models more, but not in a fixed capability order even within a family.

4.4. Aggregate Scores Conceal Capability Failures

The degradation map exposes failures that an aggregate would hide. The clearest case is code generation for small chat-tuned models. Under the standard raw-completion HumanEval protocol used for all models, SmolLM2-1.7B scores 0.6 percent at FP16 and 0.0 percent at four-bit, numbers that, at face value, suggest the model simply cannot code. Inspection of its generations shows the cause is a prompting-protocol artifact: SmolLM2 is heavily chat-tuned and, given a bare function signature to continue, emits prose and markdown rather than code, whereas the other six models continue raw code normally. When SmolLM2 is instead prompted with its chat template and the emitted function is extracted and executed, it scores 34.2 percent at FP16 and 14.6 percent at four-bit, a relative degradation of 57 percent, the largest code degradation in this study. The raw-completion protocol had hidden a near-total collapse of a real capability. Two lessons follow. First, capability-specific evaluation can be sensitive to prompting protocol, and a single near-zero score should be audited before being read as inability. Second, and more important for our thesis, the underlying phenomenon is severe: a 1.7-billion-parameter model loses more than half of a genuine code-generation ability under four-bit quantization, which no aggregate score in our suite surfaces. We report this chat-protocol result separately because it uses a different prompt format from the other models; the cross-model consistency analysis uses only the common protocol.

4.5. Discussion

Three themes emerge. First, quantization damage is concentrated in a minority of capabilities; reporting a single accuracy number is not only lossy but actively misleading when the lost capability is the one a deployment depends on, such as multilingual reasoning or code. Second, fragility is partly intrinsic to the capability: multilingual mathematics and code are fragile across architectures while

commonsense is robust, suggesting the brittleness is tied to what the capability demands, namely long multi-step generation and exact-token correctness, more than to the architecture. Third, beyond the shared extremes the ordering is architecture-dependent, so practitioners cannot port a degradation profile across families. The resulting recommendation is concrete: before shipping a quantized small model, re-test its multilingual and code capabilities specifically, and do not rely on an aggregate score or on another model's profile.

5. Conclusion

We presented a capability-level degradation map for seven small instruction-tuned LLMs under four-bit quantization. Degradation is strongly capability-specific: multilingual mathematics and code are the most fragile capabilities, with up to 57 percent relative loss, while commonsense reasoning is nearly untouched. The degradation ordering is only weakly consistent across architectures, with a mean Kendall's tau of 0.29, so the extremes are predictable but the full profile is architecture-dependent, and smaller models degrade more in magnitude. Because aggregate accuracy hides these capability-specific failures, we recommend that small-model quantization be reported and audited per capability. Future work should broaden the study to additional quantization methods, more than one benchmark per capability, and more architecture families, and should extend the analysis to multimodal small models.

References

[1] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30, pp. 5998–6008. doi: 10.5555/3295222.3295349.

[2] T. B. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 1877–1901. doi: 10.5555/3495724.3495883.

[3] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023. doi: 10.48550/arXiv.2302.13971.

[4] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," *arXiv preprint arXiv:2208.07339*, 2022. doi: 10.48550/arXiv.2208.07339.

[5] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022. doi: 10.48550/arXiv.2210.17323.

[6] J. Lin *et al.*, "AWQ: Activation-aware weight quantization for LLM compression and acceleration," *arXiv preprint arXiv:2306.00978*, 2023. doi: 10.48550/arXiv.2306.00978

[7] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," *arXiv preprint*

[8] R. Jin, J. Du, W. Huang, W. Liu, J. Luan, B. Wang, and D. Xiong, "A comprehensive evaluation of quantization strategies for large language models," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 12186–12215. doi: 10.18653/v1/2024.findings-acl.726

[9] S. Li *et al.*, "Evaluating quantized large language models," *arXiv preprint arXiv:2402.18158*, 2024. doi: 10.48550/arXiv.2402.18158.

[10] K. Marchisio, S. Dash, H. Chen, D. Aumiller, A. Üstün, S. Hooker, and S. Ruder, "How does quantization affect multilingual LLMs?," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 15928–15947. doi: 10.18653/v1/2024.findings-emnlp.935.

[11] W. Huang *et al.*, "An empirical study of LLaMA3 quantization: From LLMs to MLLMs," *arXiv preprint arXiv:2404.14047*, 2024. doi: 10.48550/arXiv.2404.14047.

[12] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," *arXiv preprint arXiv:2211.10438*, 2022. doi: 10.48550/arXiv.2211.10438.

[13] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers," *arXiv preprint arXiv:2206.01861*, 2022. doi: 10.48550/arXiv.2206.01861.

[14] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: k-bit inference scaling laws," *arXiv preprint arXiv:2212.09720*, 2022. doi: 10.48550/arXiv.2212.09720.

[15] S. Ashkboos *et al.*, "QuaRot: Outlier-free 4-bit inference in rotated LLMs," *arXiv preprint arXiv:2404.00456*, 2024. doi: 10.48550/arXiv.2404.00456.

[16] S. Kim *et al.*, "SqueezeLLM: Dense-and-sparse quantization," *arXiv preprint arXiv:2306.07629*, 2023. doi: 10.48550/arXiv.2306.07629.

[17] E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in a single forward pass," *arXiv preprint arXiv:2301.00774*, 2023. doi: 10.48550/arXiv.2301.00774.

[18] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-only information," *arXiv preprint arXiv:1712.05877*, 2017. doi: 10.48550/arXiv.1712.05877.

[19] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015. doi: 10.48550/arXiv.1510.00149.

[20] I. Hubara, M. Courbariaux, R. El-Yaniv, and Y. Bengio, "Training neural networks with low precision weights and activations," *arXiv preprint arXiv:1609.07061*, 2016. doi: 10.48550/arXiv.1609.07061.

[21] R. Banner, Y. Nahshan, and D. Soudry, "Post-training 4-bit quantization of convolutional networks for rapid-deployment," *arXiv preprint arXiv:1810.05723*, 2018. doi: 10.48550/arXiv.1810.05723.

[22] M. Nagel, R. A. Amjad, M. v. Baalen, C. Louizos, and T. Blankevoort, "Up or down? Adaptive rounding for post-training quantization," in *International Conference on Machine Learning (ICML)*, 2020, pp. 7197–7206.

[23] S. Shen, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Q-BERT: Hessian based ultra low precision quantization of BERT," *arXiv preprint arXiv:1909.05840*, 2019. doi: 10.48550/arXiv.1909.05840.

[24] Y. Bondarenko, M. Nagel, and T. Blankevoort, "Understanding and overcoming the challenges of efficient transformer quantization," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021, pp. 7947–7958. doi: 10.18653/v1/2021.emnlp-main.627.

[25] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, "A survey on model compression for large language models," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1556–1577, 2024. doi: 10.1162/tacl_a_00704.

[26] E. J. Hu *et al.*, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021. doi: 10.48550/arXiv.2106.09685.

[27] A. Yang *et al.*, "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115.

[28] A. Grattafiori *et al.*, "The Llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783.

[29] Gemma Team, M. Riviere, *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024. doi: 10.48550/arXiv.2408.00118.

[30] M. Abdin *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024. doi: 10.48550/arXiv.2404.14219..

[31] L. B. Allal *et al.*, "SmolLM2: When smol goes big -- Data-centric training of a 1.7B language model," *arXiv preprint arXiv:2502.04353*, 2025. doi: 10.48550/arXiv.2502.04353.

[32] L. Gao *et al.*, "A framework for few-shot language model evaluation," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2023, pp. 270–283.

[33] P. Clark *et al.*, "Think you have solved question answering? Try ARC, the AI2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457.

[34] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "HellaSwag: Can a machine really finish your sentence?," in *Proceedings of the 57th Annual Meeting of the*

Association for Computational Linguistics, 2019, pp. 4791–4800. doi: 10.18653/v1/P19-1472

[35] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168.

[36] F. Shi *et al.*, "Language models are multilingual chain-of-thought reasoners," *arXiv preprint arXiv:2210.03057*, 2022. doi: 10.48550/arXiv.2210.03057.

[37] M. Chen *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374.

[38] J. Zhou *et al.*, "Instruction-following evaluation for large language models," *arXiv preprint arXiv:2311.07911*, 2023. doi: 10.48550/arXiv.2311.07911..

[39] D. Hendrycks *et al.*, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2020. doi: 10.48550/arXiv.2009.03300.

[40] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022. doi: 10.48550/arXiv.2201.11903.

[41] W. Kwon *et al.*, "Efficient memory management for large language model serving with pagedattention," *arXiv preprint arXiv:2309.06180*, 2023. doi: 10.48550/arXiv.2309.06180.

[42] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, 1938. doi: 10.1093/biomet/30.1-2.81.