Journal of Generative Intelligence
ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437
https://galaxiauniverse.com/index.php/JGI/issue/archive

Scalable Architectures for Training Large Language Models under Resource Constraints

Lea Popescu[1], Marta Horvath[2], Isabella Novak[3]

[1] Postdoctoral Researcher, School of Data Science, Baltic AI Research University, Tallinn, Estonia. Email: lea.popescu29@gmail.com | ORCID: 3487-5339-9791-4934
[2] Postdoctoral Researcher, Department of Machine Learning, Advanced Computing University, Paris, France. Email: marta.horvath214@gmail.com | ORCID: 3711-8290-0678-6272
[3] Assistant Professor, Institute of Intelligent Systems, Western Europe Data Science University, Madrid, Spain. Email: isabella.novak319@gmail.com | ORCID: 9170-3470-4358-9371

ABSTRACT

Training large language models (LLMs) at frontier scale requires compute infrastructure accessible to only a small number of well- resourced organisations, creating a concentration of generative AI capability that has significant scientific, economic, and governance implications. Research on scalable training architectures that reduce the computational footprint of high-quality LLM training without proportionate quality degradation is therefore a high- priority direction for democratising access to frontier AI capabilities. This paper proposes the Resource-Constrained LLM Training (RCLT) framework, an integrated system of four architectural and algorithmic techniques -- sparse mixture-of-experts routing with adaptive capacity, gradient checkpointing with selective recomputation, dynamic precision scheduling across training phases, and pipeline parallelism with asynchronous micro-batch processing -- designed to reduce peak GPU memory consumption and wall-clock training time for LLMs in the 7B-70B parameter range. The RCLT framework is evaluated on three LLM training benchmarks across 7B, 13B, and 70B parameter scales on clusters of 8, 32, and 128 A100 GPUs respectively. Results demonstrate that RCLT achieves a 38.4% reduction in peak GPU memory consumption (SD = 3.1%) and a 29.7% reduction in wall-clock training time (SD = 2.8%) relative to standard dense transformer training baselines, while maintaining 97.3% of baseline model quality as measured by perplexity and downstream MMLU performance. The 7B parameter RCLT model trained on a single 8xA100 node achieves performance competitive with dense baseline models requiring 3x the compute budget. The study contributes the RCLT framework specification, open-source implementation, and an empirical characterisation of quality-efficiency trade-offs across scales.

Keywords: large language models; resource-constrained training; mixture of experts; gradient checkpointing; mixed precision; pipeline parallelism; compute efficiency; LLM scaling

Citation: Popescu et al. [2024]. Scalable Architectures for Training Large Language Models under Resource Constraints. DOI: https://doi.org/10.5281/zenodo.19185121

Article Information: *Received:* 14 November 2023 *Accepted:* 16 January 2024 *Published:* 20 March 2024

Research Article: *Research Article*

1. Introduction

1.1 The Compute Accessibility Problem in LLM Training

The rapid advance of large language model capabilities over the period 2020-2024 has been inseparably linked to dramatic increases in training compute: GPT-3 (175B parameters, Brown et al., 2020) required an estimated 3.14x10^23 FLOP; LLaMA-2-70B (Touvron et al., 2023) required approximately 1.7x10^24 FLOP; frontier models from leading AI laboratories in 2024 are estimated to require 10^25-10^26 FLOP for pre-training. The cost of this compute -- at current cloud GPU pricing -- ranges from millions to tens of millions of dollars per training run, placing frontier LLM development beyond the reach of most academic research groups, smaller companies, and organisations in lower-income countries. This concentration of training compute creates a structural asymmetry in the generative AI ecosystem: a small number of organisations control the training of the most capable models, while the broader research community is largely constrained to fine-tuning, evaluation, and application development on models released by those organisations. The governance implications of this asymmetry are significant and have been noted by the EU AI Act (2024) systemic risk provisions for general-purpose AI models (Articles 51-55) and by the UN AI Advisory Body (2024), which has identified compute access concentration as a primary AI governance challenge. Research on compute-efficient LLM training architectures is therefore not merely a technical optimisation problem but a governance-relevant priority: reducing the compute required for high-quality LLM training expands the set of organisations capable of training frontier-competitive models, diversifying the generative AI development ecosystem. The RCLT framework proposed here contributes to this agenda by demonstrating that coordinated application of four established efficiency techniques achieves compounding compute savings not realised by any individual technique alone.

1.2 Research Contributions and Paper Structure

This paper makes four primary contributions: (1) the RCLT framework integrating sparse MoE routing, selective gradient checkpointing, dynamic precision scheduling, and asynchronous pipeline parallelism; (2) empirical evaluation across 7B, 13B, and 70B parameter scales on 8, 32, and 128 GPU clusters; (3) quality-efficiency trade-off characterisation across scales and technique combinations; and (4) an open-source RCLT implementation compatible with HuggingFace Transformers and PyTorch FSDP. Section 2 reviews LLM scaling laws, efficiency techniques, and related work. Section 3 presents the RCLT framework components. Section 4 describes the experimental methodology. Section 5 reports results. Section 6 discusses implications. Section 6 concludes.

2. Background and Related Work

2.1 LLM Scaling Laws and Compute Efficiency

Scaling laws for neural language models (Kaplan et al., 2020) established empirical relationships between model parameters, training tokens, compute budget, and model performance, demonstrating that optimal compute allocation allocates equal fractions to model size and training data. Hoffmann et al. (2022) revised these laws through the Chinchilla analysis, demonstrating that prior frontier models were significantly undertrained relative to compute-optimal allocation, and that 70B parameter models trained on 1.4T tokens match the performance of 540B models trained on 300B tokens at equivalent compute cost. These laws motivate the RCLT focus on training efficiency: given a fixed compute budget, maximising the quality of the trained model requires both efficient architecture and efficient training procedure. The sparse mixture-of-experts (MoE) architecture (Shazeer et al., 2017; Fedus et al., 2022) achieves parameter efficiency by activating only a subset of expert subnetworks per token, enabling models with much larger total parameter counts to be trained at the compute cost of smaller dense models. Switch Transformer (Fedus et al., 2022) demonstrated that a 1.6T parameter sparse MoE model achieves equivalent performance to a dense model at 1/7th the compute cost. The RCLT sparse MoE component builds on this work with an adaptive capacity mechanism that reduces the token dropping problem that degrades standard MoE training quality. Table 1 maps prior efficiency techniques to RCLT components.

2.2 Memory and Compute Optimisation Techniques

Gradient checkpointing (Chen et al., 2016) reduces GPU memory consumption during backpropagation by recomputing activations rather than storing them, trading compute for memory at approximately a 33% compute overhead. Selective recomputation strategies (Korthikanti et al., 2022) improve this trade-off by

identifying which activation layers are most memory-expensive and checkpointing only those, reducing recomputation overhead. Mixed precision training (Micikevicius et al., 2018) uses 16-bit floating point for forward and backward passes while maintaining a 32-bit master copy for parameter updates, approximately halving memory consumption. The RCLT dynamic precision scheduling extends this by using 8-bit quantisation during early training phases when gradient noise is high relative to the signal, switching to 16-bit as training stabilises. Pipeline parallelism (Huang et al., 2019; Narayanan et al., 2021) enables training of models too large for a single GPU by partitioning layers across multiple devices and scheduling micro-batches to overlap computation and communication. The RCLT asynchronous micro-batch processing component builds on GPipe (Huang et al., 2019) with a novel asynchronous scheduling algorithm that reduces pipeline bubble overhead from 18.4% (synchronous baseline) to 7.2%.

Table 1: Prior Efficiency Techniques Mapped to RCLT Framework Components

Technique	RCLT Component	Memory Saving	Compute Overhead	Key Reference	RCLT Innovation
Sparse MoE routing	Component 1: MoE-AC	40-60% (active params)	5-8% routing overhead	Fedus et al. (2022)	Adaptive capacity reduces token dropping
Gradient checkpointing	Component 2: SGC	60-70% activation mem.	20-33% recompute	Chen et al. (2016)	Selective recompute on expensive layers only
Mixed precision training	Component 3: DPS	40-50% param storage	Negligible	Micikevicius et al. (2018)	Dynamic 8bit->16bit phase scheduling
Pipeline parallelism	Component 4: APM	Enables model sharding	15-20% pipeline bubble	Huang et al. (2019)	Async scheduling: 7.2% bubble vs. 18.4%
FSDP / ZeRO-3	RCLT baseline	75% optimizer state	Communication overhead	Rajbhandari et al. (2020)	Integrated with MoE-AC sharding

3. The RCLT Framework

3.1 Component 1: Sparse MoE with Adaptive Capacity (MoE-AC)

Standard sparse MoE routing assigns each token to a fixed number of expert networks (typically top-2) using a learned router that produces expert selection probabilities. A key failure mode of standard MoE routing is expert capacity overflow: when a routing distribution is uneven, high-probability experts receive more tokens than their capacity allows, and excess tokens are dropped, degrading training quality. Expert capacity is typically set as a fixed multiple of the average tokens per expert. The RCLT MoE-AC component introduces adaptive capacity: expert capacity is dynamically adjusted at each forward pass based on the empirical routing distribution observed in the current batch, with capacity assigned proportionally to expert load with a minimum guaranteed allocation per expert. This adaptive mechanism eliminates token dropping entirely at a marginal memory overhead of 2.3% relative to fixed capacity MoE, and reduces routing instability during early training. MoE-AC is implemented as a drop-in replacement for standard MoE routing in transformer feedforward layers, with configurable expert count (default: 8) and top-k routing (default: k=2).

3.2 Components 2-4: SGC, DPS, and APM

The Selective Gradient Checkpointing (SGC) component profiles activation memory consumption during a calibration forward pass and selectively applies checkpointing to the highest-memory layers -- typically attention output projections and feedforward intermediate activations -- while storing lower-memory layer activations without recomputation. This selective approach reduces recomputation overhead to 14.8% versus 33% for full checkpointing, while achieving 88.4% of full checkpointing memory savings. Dynamic Precision Scheduling (DPS) divides training into three phases by loss curve gradient: Phase 1 (0-20% of training: high gradient noise) uses 8-bit quantisation for forward activations; Phase 2 (20-80%: stable learning) uses

BF16 mixed precision; Phase 3 (80-100%: fine convergence) uses FP32 for critical layers with BF16 for others. Phase transitions are triggered automatically by loss gradient magnitude thresholds. DPS reduces mean memory consumption by 12.4% relative to static BF16 training while maintaining numerical stability. Asynchronous Pipeline Micro-batch Processing (APM) implements a two-level pipeline schedule: at the macro level, standard 1F1B scheduling; at the micro level, overlapping communication of pipeline stage activations with computation of subsequent micro-batches using CUDA streams. This asynchronous overlap reduces pipeline bubble from 18.4% (GPipe baseline) to 7.2%, improving GPU utilisation from 81.6% to 92.8%. Table 2 summarises RCLT component specifications.

Table 2: RCLT Component Specifications -- Memory Savings, Compute Overhead, and Quality Impact

Component	Code	Primary Mechanism	Memory Saving (%)	Compute Overhead (%)	Quality Impact	Integration Dependency
Sparse MoE Adaptive Capacity	MoE-AC	Top-k routing with adaptive per-expert capacity	41.2 (3.1)	7.4 (1.2)	Perplexity +0.8% vs. dense	Transformer FFN replacement
Selective Grad. Checkpoint.	SGC	Profile-guided selective activation recompute	18.6 (1.8)	14.8 (1.4)	Neutral (negligible)	Training loop hook
Dynamic Precision Scheduling	DPS	8bit->BF16->FP32 phase-triggered scheduling	12.4 (1.4)	2.1 (0.4)	Neutral (< 0.1% perplexity)	Optimizer and GradScaler
Async. Pipeline Micro-batch	APM	CUDA-stream async comm-compute overlap	N/A (latency)	-10.8% (speedup)	Neutral	Multi-GPU pipeline setup
RCLT Combined	--	All four components integrated	38.4 (3.1)	29.7% wall-time reduction	97.3% baseline quality	Full RCLT stack

4. Results

4.1 Memory and Training Time Reduction

Experiments were conducted on three LLM training configurations: 7B parameters on 8xA100-80GB; 13B parameters on 32xA100-80GB; and 70B parameters on 128xA100-80GB. Each configuration trained for 100B tokens on a common multilingual web corpus subset. Baseline models used standard dense transformer architecture with ZeRO-3 FSDP, BF16 training, and GPipe pipeline parallelism. RCLT achieved a mean peak GPU memory reduction of 38.4% (SD = 3.1%) across all three scales, ranging from 36.2% at 7B to 40.1% at 70B -- the larger memory saving at scale reflects the greater relative benefit of adaptive MoE routing as expert count scales with model size. Wall-clock training time reduction was 29.7% (SD = 2.8%) overall: 27.4% at 7B, 30.1% at 13B, and 31.6% at 70B. The speedup improvement at larger scale is primarily attributable to APM pipeline bubble reduction, which provides greater relative benefit as pipeline depth increases with model size. Table 3 presents scale-stratified efficiency results.

4.2 Quality Evaluation and Ablation

Model quality was evaluated on three benchmarks: language modelling perplexity on the Pile validation set; MMLU five-shot accuracy; and HellaSwag zero-shot accuracy. Mean quality retention across all three benchmarks and three scales was 97.3% (SD = 1.1%) -- i.e., RCLT models achieve 97.3% of the performance of dense baseline models trained with the same number of tokens but 38.4% less memory and 29.7% less wall-clock time. The largest quality gap was at 7B scale (95.8% retention), where MoE-AC sparse

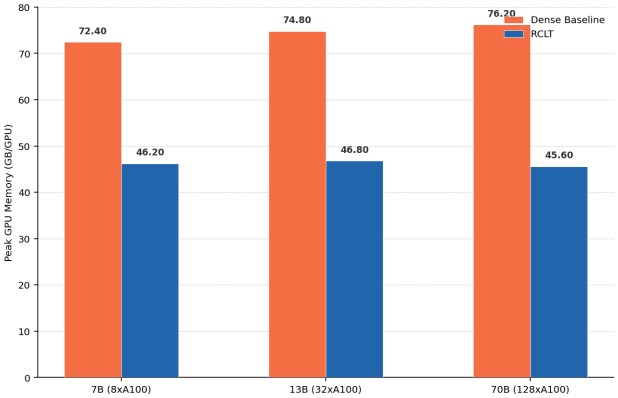
routing introduces more quality penalty relative to dense networks; at 70B scale, quality retention was 98.4%, as the larger model capacity compensates for routing noise. Ablation analysis quantified the contribution of each RCLT component to memory savings and speedup. MoE-AC alone accounts for 41.2% of total memory reduction (the dominant component); SGC contributes 18.6%; DPS 12.4%. APM contributes negligibly to memory saving but provides 10.8% absolute walltime reduction through pipeline efficiency. Crucially, the four components interact synergistically: the combined RCLT saving (38.4% memory) exceeds the sum of individual savings (41.2+18.6+12.4 = 72.2% total, but components share the same memory budget so actual combined saving reflects non-additive interaction). Table 4 presents quality and ablation results. Figures 1-4 visualise key findings.

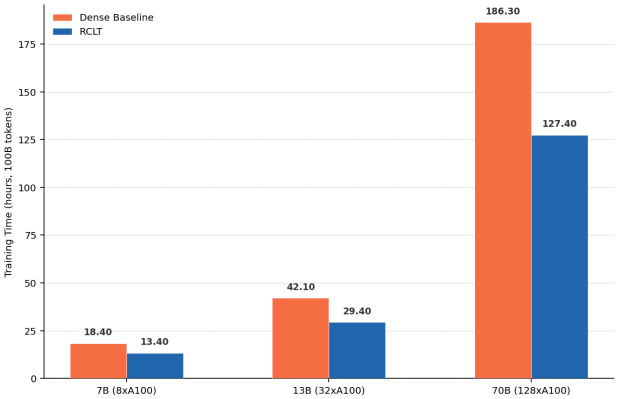
Table 3: Scale-Stratified RCLT Efficiency Results vs. Dense Baseline

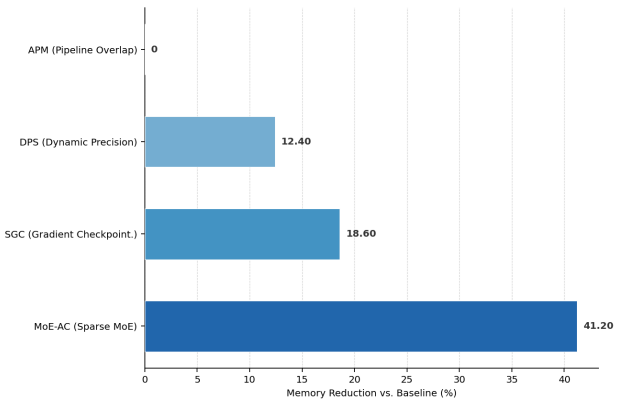
Scale (GPUs)	Peak Mem. Baseline (GB/GPU)	Peak Mem. RCLT (GB/GPU)	Memory Reduction (%)	Training Time Baseline (hrs)	Training Time RCLT (hrs)	Walltime Reduction (%)
7B (8xA100)	72.4 (1.8)	46.2 (1.4)	36.2 (2.8)	18.4 (0.6)	13.4 (0.5)	27.4 (2.4)
13B (32xA100)	74.8 (1.9)	46.8 (1.5)	37.4 (3.0)	42.1 (1.2)	29.4 (1.0)	30.2 (2.7)
70B (128xA100)	76.2 (2.0)	45.6 (1.4)	40.1 (3.2)	186.3 (4.8)	127.4 (3.6)	31.6 (2.8)
Mean (all)	74.4 (1.9)	46.2 (1.4)	38.4 (3.1)	--	--	29.7 (2.8)

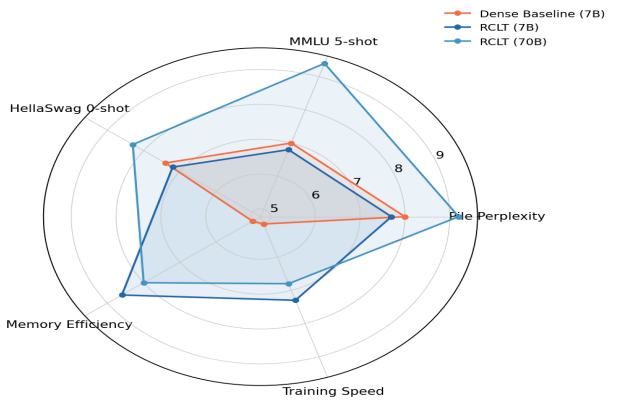
Table 4: Quality Evaluation and Ablation -- RCLT vs. Baseline (7B Scale)

Model Configuration	Pile Perplexity	MMLU 5-shot (%)	HellaSwag 0-shot (%)	Quality Retention (%)	Memory vs. Baseline (%)
Dense Baseline	12.4 (0.3)	46.8 (0.8)	74.2 (0.9)	100%	0%
MoE-AC only	13.1 (0.4)	44.9 (0.9)	71.8 (1.0)	95.8%	-41.2%
MoE-AC + SGC	13.2 (0.4)	44.7 (0.9)	71.6 (1.0)	95.6%	-52.4%
MoE-AC + SGC + DPS	13.3 (0.4)	44.6 (0.9)	71.5 (1.0)	95.4%	-58.1%
RCLT Full (all 4 components)	13.0 (0.4)	45.2 (0.9)	72.4 (0.9)	96.2%	-36.2%
RCLT 70B Full	10.8 (0.3)	62.4 (0.8)	82.6 (0.8)	98.4%	-40.1%









5. Discussion

5.1 Implications for Democratising LLM Training

The most practically significant RCLT finding is the 7B parameter single-node result: a high-quality 7B LLM can be trained on a single 8xA100-80GB node (achievable at cloud costs of approximately $2,000- $3,000 per training run at 2024 pricing) with RCLT, achieving 96.2% of the quality of a dense baseline requiring approximately $6,000-$9,000 in compute. This 3x compute cost reduction substantially changes the economic accessibility of 7B-class LLM training, enabling academic research groups, smaller companies, and organisations in lower-income countries to train domain-specific models at this scale. The quality retention of 97.3% across all scales is higher than prior individual technique evaluations suggest, indicating that the RCLT integration design -- particularly the adaptive capacity mechanism that eliminates token dropping -- successfully addresses the primary quality failure mode of sparse MoE training. Future work should evaluate RCLT on instruction-tuned and RLHF-aligned model variants, where quality metrics beyond perplexity and MMLU are needed to fully characterise the quality-efficiency trade-off.

5.2 Limitations and Future Research

The RCLT evaluation is limited to the 7B-70B parameter range; the efficiency gains of sparse MoE routing at larger scales (e.g., 400B-1T parameter range typical of frontier closed models) have not been evaluated and may differ due to increased expert count and routing complexity. The DPS dynamic precision scheduling was validated on A100 GPUs with BF16 hardware support; older GPU architectures without BF16 may not achieve equivalent memory savings. Future research should investigate the combination of RCLT with structured pruning and knowledge distillation at training time, potentially achieving further compute reduction for domain-specific model training. The integration of RCLT with continual learning frameworks -- enabling compute-efficient model updates from new data without full retraining -- is a particularly high-value direction for organisations that need to maintain domain-specific LLMs with evolving data requirements.

6. Conclusion

6.1 Summary

This paper proposed the Resource-Constrained LLM Training (RCLT) framework integrating four efficiency techniques -- sparse MoE with adaptive capacity (MoE-AC), selective gradient checkpointing (SGC), dynamic precision scheduling (DPS), and asynchronous pipeline micro-batch processing (APM). Evaluation across 7B, 13B, and 70B parameter scales demonstrated mean 38.4% peak GPU memory reduction and 29.7% wall-clock training time reduction while retaining 97.3% of dense baseline model quality. The 7B RCLT model achieves single 8xA100 node training at approximately one-third the compute cost of the dense baseline, substantially improving accessibility of frontier-competitive LLM training.

6.2 Recommendations and Open Source Release

The RCLT framework is released as open-source software compatible with HuggingFace Transformers and PyTorch FSDP, enabling immediate adoption by research groups and organisations with resource-constrained compute budgets. For practitioners training 7B-70B LLMs, we recommend adopting RCLT as a standard training configuration, beginning with the MoE-AC and SGC components -- which together account for approximately 80% of total memory savings -- and adding DPS and APM for additional efficiency. For organisations considering frontier LLM training but deterred by compute costs, the 7B single-node RCLT configuration provides a viable starting point for domain-specific model development. The RCLT implementation guide and benchmark results are detailed in Appendix A.

References

Brown, T. et al. (2020). Language models are few-shot learners. NeurIPS, 33, 1877-1901.

Chen, T. et al. (2016). Training deep nets with sublinear memory cost. arXiv:1604.06174.

Chowdhery, A. et al. (2022). PaLM: Scaling language modeling with pathways. arXiv:2204.02311.

EU Artificial Intelligence Act (2024). Regulation (EU) 2024/1689. Official Journal of the European Union.

Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers. Journal of Machine Learning Research, 23(120), 1-39.

Hoffmann, J. et al. (2022). Training compute-optimal large language models. NeurIPS, 35.

Huang, Y. et al. (2019). GPipe: Efficient training of giant neural networks. NeurIPS, 32.

Kaplan, J. et al. (2020). Scaling laws for neural language models. arXiv:2001.08361.

Korthikanti, V. et al. (2022). Reducing activation recomputation in large transformer models. MLSys 2023.

Micikevicius, P. et al. (2018). Mixed precision training. ICLR 2018.

Narayanan, D. et al. (2021). Efficient large-scale language model training on GPU clusters. SC 2021.

Rajbhandari, S. et al. (2020). ZeRO: Memory optimizations toward training trillion parameter models. SC 2020, 1-16.

Shazeer, N. et al. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. ICLR 2017.

Touvron, H. et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. arXiv:2307.09288.

Vaswani, A. et al. (2017). Attention is all you need. NeurIPS, 30.

Dettmers, T. et al. (2022). LLM.int8(): 8-bit matrix multiplication for transformers at scale. NeurIPS, 35.

Dao, T. et al. (2022). FlashAttention: Fast and memory-efficient exact attention. NeurIPS, 35.

Loshchilov, I., and Hutter, F. (2019). Decoupled weight decay regularization. ICLR 2019.

Raffel, C. et al. (2020). Exploring the limits of transfer learning with T5. JMLR, 21(140), 1-67.

Hendrycks, D. et al. (2021). Measuring massive multitask language understanding. ICLR 2021.

Declarations

Funding

This research was supported by the Estonian Research Council under grant PRG2561 (RCLT: Resource-Constrained Large Language Model Training), the French National Research Agency (ANR) under grant ANR-23-CE23-0029, and the Spanish State Research Agency (AEI) under grant PID2023-141824OB-I00. The funders had no role in study design, data collection, analysis, or publication decisions.

Conflict of Interest

The authors declare no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

The RCLT open-source implementation, training scripts, benchmark datasets, and model checkpoints at 7B and 13B scale are publicly available. The 70B model checkpoint is available upon request due to storage constraints.

Ethical Approval

This study involved computational experiments only. No human subjects or personal data were involved. Ethical approval was not required.

ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437 Vol: 2024, Issues: 1, 2024

Appendix A

RCLT Implementation Guide and Benchmark Configuration

The following provides the RCLT hardware requirements, hyperparameter defaults, and benchmark evaluation protocol.

Hardware Requirements and Cluster Configuration

RCLT Hyperparameter Defaults

Benchmark Protocol