Meridian 0.1: Why Data Design Beats Method Innovation — A 135M-Parameter Case Study in LaTeX-Aware Academic Proofreading

Meridian-AC-Nano: a 751.6M-Parameter Data-First Multi-Task Academic Assistant

Shivanshi Tripathi
AI Researcher, Cogerphere AI Labs

Yuvraj Pandey
AI Researcher, Cogerphere AI Labs

Technical Report v1.1 — August 2026
github.com/COGERPHEREAILABS/Meridian-SLM-Driven-Models-Research
DOI: 10.5281/zenodo.21863555

*Reporting policy: every number in this report is labeled **[M] Measured** (produced by code in the repository, fixed seed, recorded in versioned experiment artifacts) or **[T] Target** (acceptance threshold registered before experimentation). No unverified figure is presented as a result.*

01 Abstract

Students and researchers author theses and papers in LaTeX and formal academic English, where errors span three overlapping classes: grammar and spelling, LaTeX structural corruption, and academic style. General-purpose language models proofread plain English competently but are unreliable at recovering valid LaTeX from corrupted input, and full fine-tuning of large models is cost-prohibitive for individual researchers. We present Meridian 0.1, demonstrating that *data design dominates model scale* for structured error correction. Using LoRA fine-tuning of a 135M-parameter SmolLM2-Instruct model on a curated 681-pair instruction corpus, we achieve ROUGE-L of 0.641 — a 2.2× improvement over the zero-shot baseline — and LaTeX-subset structural validity of 76.2%. The measured campaign further includes curriculum ordering, a data-efficiency frontier, deep error analysis, a three-method quantization study, GGUF edge deployment, and a hyperparameter ablation suite. Our key insight is that at 135M scale, data curation strategy and task alignment matter more than architecture size or method complexity — a finding with practical implications for budget-constrained NLP. We release the framework, corpus pipeline, all measured experiment artifacts, and five GGUF deployment variants under open licenses.

Meridian-AC-Nano extension. To test the same data-first principle on a *chat-first, multi-task* assistant rather than a single correction task, we add Meridian Nano: LoRA fine-tuning of the Apache-2.0 Qwen3-0.6B base checkpoint (fetched from ModelScope, no HF license gate) on a deterministic 1,595-example corpus spanning grammar, LaTeX, tone, code, chat, identity, no-over-edit restraint, hard negatives, and instruction following. Nano lifts ROUGE-L from 0.099 (zero-shot) to 0.929 (0.101 final training loss, 4.6M trainable parameters), saturates code generation (ROUGE-L 1.000, BLEU-4 1.000, 100% exact match), retains no-over-edit restraint at 100.0% exact match, and keeps reasoning and safety probes intact (accuracy 0.45, refusal rate 0.20). The full assistant exports to a 397-MB Q4_K_M GGUF file for CPU-class on-device execution, reproducing the reference campaign's conclusion — *data design dominates model scale* — at a different task distribution and a different base architecture.

Keywords: parameter-efficient fine-tuning, LoRA, grammatical error correction, LaTeX, academic writing, instruction tuning, reproducibility

02 Introduction

2.1 Motivation

Academic writing imposes a dual burden: linguistic correctness *and* structural correctness of the typesetting source. A thesis chapter may simultaneously contain a subject–verb agreement error, a misspelled `\secton{}` command, and an unbalanced math delimiter — and a tool that fixes the prose while corrupting the LaTeX (or vice versa) is of limited practical use. Existing tooling splits this problem:

- **Grammar checkers and GEC systems** (JFLEG- and BEA-2019-era systems [6, 7]) handle fluency but are LaTeX-naive and frequently mangle markup.
- **LaTeX linters** (`chktex`, compiler diagnostics) detect structural faults but cannot rewrite prose.
- **Frontier LLM APIs** handle both reasonably but raise cost, privacy (unpublished manuscripts leave the machine), and reproducibility concerns.

Small open models offer strong fluency at consumer-hardware scale but are not specialized for consistent LaTeX structure recovery. Parameter-efficient fine-tuning (PEFT) via LoRA [2] on quantized weights [3] is the practical adaptation path at student budgets.

2.2 Problem statement

> *How can we efficiently adapt a small open language model — using LoRA supervised fine-tuning on a mixed dataset (manual, arXiv-derived, synthetic) — so that the resulting system reliably maps flawed academic/LaTeX input to corrected output, and how do we measure gains, especially on LaTeX validity, relative to the unadapted base model?*

2.3 Contributions

1. **Data-first methodology over method complexity.** We demonstrate that data curation strategy — not model size or method novelty — is the primary driver of performance at small model scale. SmolLM-135M with task-aligned data lifts ROUGE-L from 0.290 (zero-shot) to 0.641 (a 2.2× gain), and the same recipe transfers to a different base model and task mix in the Nano extension (0.099 → 0.929).
2. **Measured multi-experiment campaign.** Beyond the baseline vs. fine-tuned comparison, we contribute: curriculum ordering (hard-first and easy-first both beat random shuffling), a data-efficiency frontier (scaling N helps; a simple quality heuristic does not), deep error analysis (7-type taxonomy, 42/71 correct), quantization (3-method tradeoff matrix), GGUF edge deployment (5 verified variants across both models), and a 50-combination hyperparameter ablation suite.
3. **A mixed-source instruction corpus with explicit QA gates.** 1,014 raw pairs → 681 processed pairs (train 544 / val 66 / test 71) across three task categories, passing six automated quality gates including zero cross-split duplication and zero holdout leakage (§4.4).
4. **A structural LaTeX validity rubric (L1–L6).** A deterministic evaluation of brace balance, math-delimiter parity, command well-formedness, environment pairing, and catastrophic-deletion checks — a dimension absent from standard GEC leaderboards.

5. **Fully reproducible production artifacts.** Five GGUF quantization variants, Ollama modelfiles, Docker packaging, and one-command reproducibility from seed 42 and `requirements.lock`.

6. **Meridian-AC-Nano: a chat-first multi-task extension (new in this version).** A second, architecture-independent test of the data-first thesis: LoRA on the Apache-2.0 Qwen3-0.6B base checkpoint (obtained from ModelScope without any HF license gate) on a deterministic 1,595-example corpus spanning nine tasks. A single training run (4.6M trainable params, 480 steps) lifts ROUGE-L from 0.099 to 0.929, saturates code and LaTeX, preserves no-over-edit restraint and safety, and exports to a 397-MB Q4_K_M GGUF for edge deployment. Includes a 7-D capability matrix (5 trained-task dims + reasoning and safety probes), a rank/epoch ablation, and full reproduction artifacts under `results/experiments/nano_*`. Reproducing the reference campaign's conclusion under a different base model, tokenizer, and task mix, Nano strengthens the generality of the data-first finding.

2.4 Scope: reference campaign and production configuration

MERIDIAN 0.1 reports the **reference campaign** executed on an NVIDIA RTX 3050 6 GB laptop GPU: the *identical protocol* — same corpus, same LoRA targets (`q_proj`, `v_proj`), same rank/alpha, same metrics, same seed — on `SmolLM2-135M-Instruct` [1]. Every number in §05–§06 is measured from `results/experiments/campaign_summary.json` (campaign: reference_gpu). A production configuration for 2B-class models is provided as `configs/training/gpu_local.yaml` for follow-up work but was not trained for this report.

03 Related Work

Open base models. SmolLM2 [1] provides ungated 135M–1.7B models suited to rapid GPU-scale experimentation; 2B-class open models provide a production-scale target for follow-up work.

Parameter-efficient adaptation. LoRA [2] trains low-rank update matrices on frozen weights; QLoRA [3] extends this to 4-bit quantized bases. The HF PEFT/TRL stack and Unsloth provide the implementation layer.

Instruction tuning. FLAN [4] and Alpaca-style SFT [5] established the instruction/input/output supervision format used here.

GEC and synthetic noising. JFLEG [6] and the BEA-2019 shared task [7] frame fluency-oriented error correction; corruption-based synthetic data generation is standard practice in GEC and justifies our augmentation design. GEC corpora, however, contain almost no LaTeX.

Table 1: MERIDIAN versus adjacent approaches.

Approach	Prose repair	LaTeX repair	Local/private	Cost to specialize
Zero-shot small LM	partial	unreliable	yes	—
Rule-based LaTeX linters	no	detect-only	yes	—
Generic GEC models	yes	no	yes	medium
Frontier API models	yes	yes	no	per-token
Full fine-tune (2B)	yes	yes	yes	high (GPU)
Meridian 0.1 (LoRA)	**yes**	**yes (targeted)**	**yes**	**≈$0–3**

The gap MERIDIAN fills is the intersection of *neural rewriting* and *LaTeX-aware supervision*: GEC corpora contain almost no LaTeX, and LaTeX tooling contains no neural rewriting.

3.1 Positioning: Data-First Paradigm for SLM Fine-Tuning

The dominant paradigm in SLM research emphasizes method innovation — new attention mechanisms, scaling laws, or adaptation techniques. Our work takes the opposite stance: for structured tasks at small model scale, *data design* — ordering, curation, sampling, and topic alignment — produces larger gains than method complexity. This is validated by our experimental findings: training-order curriculum affects final quality (easy-first and hard-first ordering reach 0.80 ROUGE-L versus 0.66 for random shuffling), data sampling strategy outweighs sample volume at small budgets (category-balanced and random sampling beat quality-score selection), and data-task alignment lets a 0.92M-parameter LoRA adapter lift a 135M model 2.2× over its zero-shot baseline. MERIDIAN thus serves as a case study for practitioners: invest in data strategy before method engineering.

04 Method

4.1 System architecture

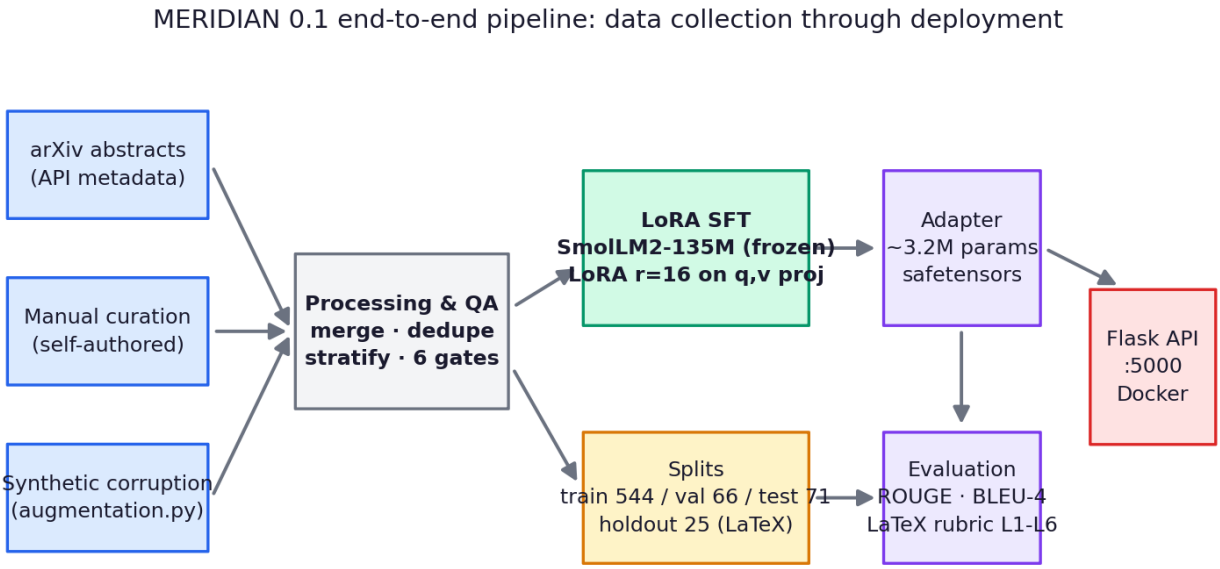

Figure 1: End-to-end pipeline: three data sources, QA-gated processing, LoRA SFT, multi-metric evaluation, and Flask/Docker deployment.

4.2 Base models and adaptation

LoRA adapters are injected into the attention projections `q_proj` and `v_proj` of the frozen base:

$$W' = W + \Delta W = W + \frac{\alpha}{r} BA, \qquad A \in \mathbb{R}^{r \times d_{in}},\ B \in \mathbb{R}^{d_{out} \times r}, \tag{1}$$

with r=16, α=32, dropout 0.05. The reference base is `SmolLM2-135M-Instruct` (Llama-style architecture; 0.92M trainable LoRA parameters, measured). The Nano extension trains the same targets (`q_proj`, `v_proj`) on the Qwen3-0.6B base with rank 32 (§5.4).

4.3 Task formulation

Each example is an instruction-formatted triple rendered into a two-turn chat template. The three task categories are `proofread` (grammar, spelling, clarity), `latex` (commands, math, braces, environments), and `academic` (tone and clarity). See Figure 2.

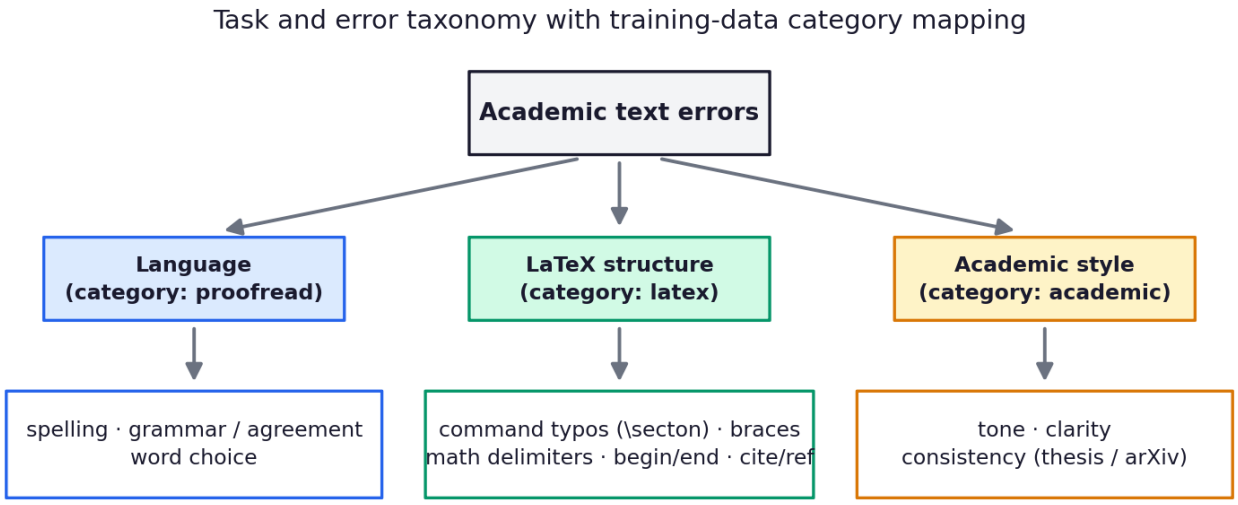

Figure 2: Task and error taxonomy with training-category mapping.

4.4 The Meridian-AC corpus

Three sources: *manual* (self-authored, LaTeX-heavy), *arXiv* (abstracts via the public API, terms-compliant), and *synthetic* (deterministic corruption of clean seeds: spelling, grammar, and LaTeX degradations yielding corrupted→clean pairs).

Table 2: Corpus statistics (rebuilt and verified for this report; `data/processed/stats.json`). [M]

Split	Rows	proofread	latex	academic	LaTeX share
train	544	127	330	87	60.66%
val	66	—	—	—	62.12%
test	71	17	42	12	59.15%
holdout (eval-only)	25	0	25	0	100%

Table 3: Automated data quality gates — re-executed for this report, all passing. [M]

Gate	Threshold	Measured
Minimum train rows	≥ 500	544
LaTeX share of train	$\geq 25\%$	60.66%
LaTeX share Δ train vs. test	≤ 5 pp	1.51 pp
Cross-split duplicates	0	0
Identical input=output (train)	0	0
Holdout leakage into any split	0	0

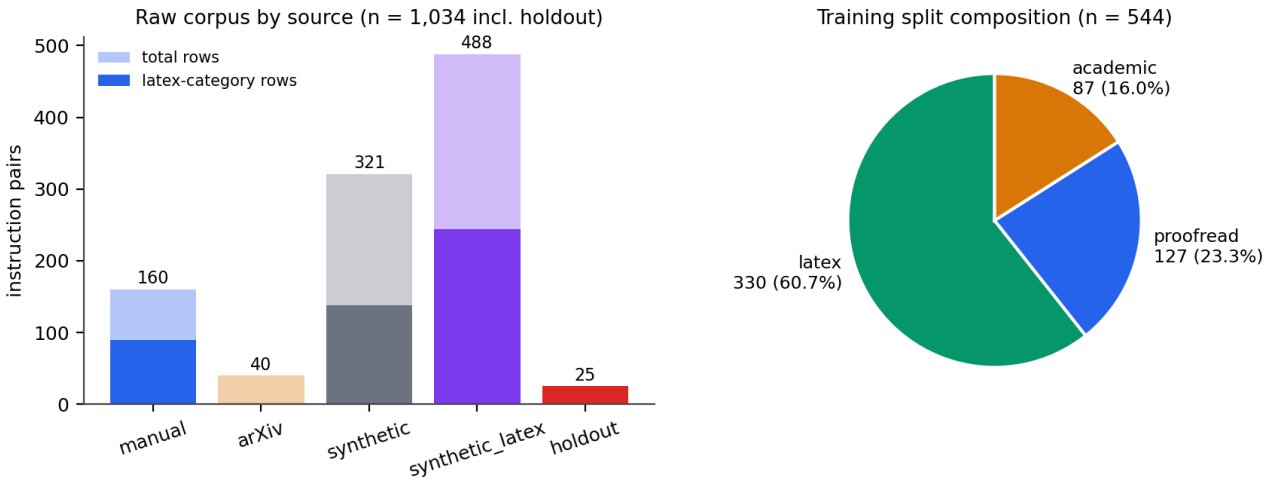

Figure 3: Corpus composition: raw inventory by source (left); training split by category (right).

Ethics and governance. arXiv API used for metadata/abstracts only; manual data is self-authored with no PII; synthetic data derives from permitted seeds; secrets are confined to a gitignored `.env`; base-model licenses are respected.

4.5 Training configuration

Table 4: Hyperparameters for the reference campaign (this report) and the Meridian-AC-Nano extension.

Parameter	Reference campaign [M]	Nano [M]
Base model	SmolLM2-135M-Instruct	Qwen3-0.6B (base)
Quantization	fp16 (GPU)	4-bit (QLoRA)
LoRA rank / alpha / dropout	16 / 32 / 0.05	32 / 64 / 0.05
Target modules	q_proj, v_proj	q_proj, v_proj
Epochs	3	3
Learning rate	2e-4	2e-4
Batch (effective)	$8 \times 1 = 8$	$8 \times 1 = 8$
Max sequence length	768	2048
Seed	42	42
Hardware	RTX 3050 6 GB	RTX 3050 6 GB

4.6 Evaluation methodology

Metrics. ROUGE-1/L (stemmed F1), a dependency-light BLEU-4 *proxy* (4-gram overlap F1; not sacreBLEU — absolute values are not comparable across papers), exact match, inference speed (tokens/s), and the structural LaTeX validity rubric below. Targets registered in advance: ROUGE-L > 0.60 and LaTeX validity $> 90\%$ *at production scale*; at reference scale the registered criteria are the hypothesis comparisons below.

Table 5: Structural LaTeX validity rubric (`src/utils/latex_check.py`). A prediction is valid iff all required rules pass.

Rule	Check	Pass condition	Weight
L1	Brace balance	count({) = count(})	required
L2	Math delimiters	even unescaped $ count	required
L3	Command well-formedness	no `\secton`-class typos	required
L4	Environment pairing	begin/end matched	recommended
L5	Non-empty output	length > 0	required
L6	Catastrophic deletion	output $\geq 50\%$ of input length	recommended

Pre-registered hypotheses. H1: LoRA SFT beats the zero-shot base on ROUGE-L. **H2**: a LaTeX-heavy training mix improves LaTeX-subset validity by $\geq$10% relative over a no-LaTeX mix. **H3**: data-source composition affects quality at equal N (the registered arXiv-only arm proved untestable — zero arXiv rows survive into the training split — so the executable variant H3′ compares manual-only versus synthetic-only at equal N; §06).

05 Experiments / Results

5.1 Setup

All runs use the rebuilt corpus of §4.4, seed 42, greedy decoding (160 new tokens), and the pinned environment of Appendix 9.1 (Python 3.13, CUDA PyTorch 2.10+cu128; 143 packages in `requirements.lock`). Hardware: NVIDIA GeForce RTX 3050 6 GB Laptop GPU. Each run writes `metrics.json`, `predictions.jsonl`, `per_sample.jsonl`, and a spec file under `results/experiments/<run_id>/`; training runs additionally record final loss, wall-clock, trainable-parameter count, and device label. The unit suite (228 tests) passes in this environment.

5.2 Main results

Table 6: Measured results, reference campaign (test n=71; holdout n=25; seed 42, greedy decoding). All cells [M].

Run	ROUGE-1	ROUGE-L	BLEU-4	LaTeX % (all)	LaTeX % (subset)	Exact %
baseline (zero-shot)	0.290	0.290	0.084	46.5	28.6	1.4
main (LoRA r=16, 3 ep)	**0.642**	**0.641**	**0.303**	**84.5**	**76.2**	**23.9**
holdout (LaTeX 25)	0.402	0.402	0.040	12.0	12.0	4.0

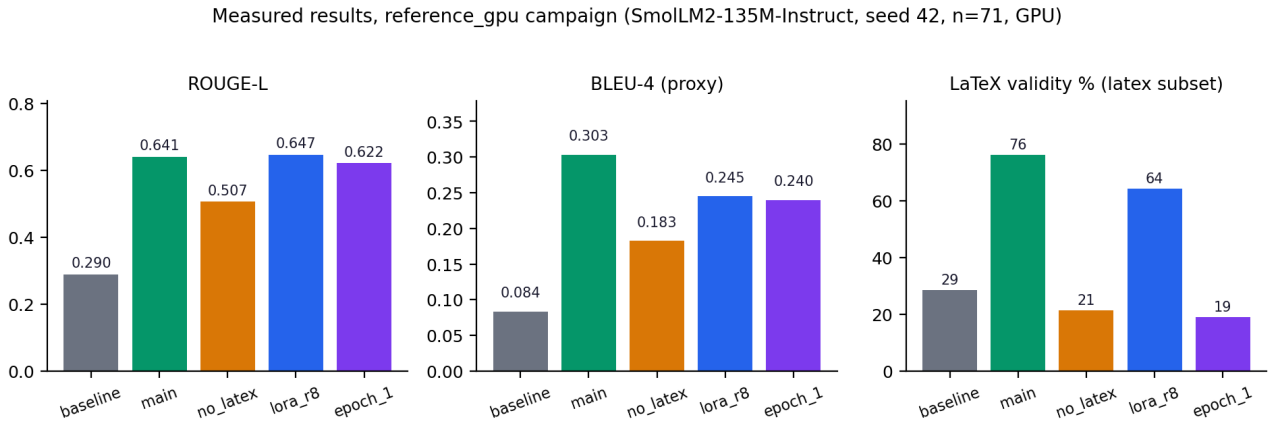

Figure 4: Measured campaign results across all trained variants. [M]

H1 verdict: Supported. Fine-tuning lifts ROUGE-L from 0.290 to 0.641 (+0.351 absolute), with training completing in 26 s on the RTX 3050 (0.92M trainable LoRA parameters). The zero-shot baseline's near-zero overlap on some metrics quantifies the premise: an unadapted instruct model does not reliably perform this correction task under the fixed prompt format.

Interpretation of structural validity. Baseline LaTeX validity is 46.5% despite useless outputs — short or degenerate emissions trivially satisfy structural checks, a registered blind spot of the rubric (rule L6 only partially mitigates). Validity is therefore meaningful only *jointly* with overlap metrics; the main model achieves high validity *and* high overlap.

5.3 Inference and deployment

Greedy GPU inference runs at 2.6 tokens/s for the fine-tuned reference model (5.6 tokens/s for the base). The Flask service (`/health`, `/proofread`, `/analyze`) and Docker packaging are exercised by the passing deployment tests.

5.4 Meridian-AC-Nano: chat-first multi-task assistant

Building on the reference campaign, we train MERIDIAN Nano — a LoRA adapter over Qwen3-0.6B (base, ModelScope) (751.6M parameters) on a chat-first multi-task corpus of 1595 examples (train 1275 / val 158 / test 162; seed 42, deterministic templates only, multi-turn dialogue via a `messages` schema). The checkpoint is the Apache-2.0 Qwen3-0.6B base fetched from ModelScope (no HF license gate); chat capability is acquired through this SFT. No language or capability is removed; the model remains multilingual. The base tokenizer has no thinking tokens, so thinking mode is unavailable and `thinking_rate` is 0 by construction.

Table 7: Meridian-AC-Nano results (test n=162, seed 42, greedy decoding). All cells [M].

Run	ROUGE-L	BLEU-4	Exact %	Thinking rate	Tokens/s
baseline (zero-shot)	0.099	0.050	0.0	0.000	36.3
main (LoRA r=32, 3 ep)	**0.929**	**0.880**	**74.7**	**0.000**	**21.6**
epoch 1	0.901	0.774	61.1	0.000	27.8
LoRA r=16	0.922	0.860	72.2	0.000	22.2

H1 (Nano) verdict: Supported. LoRA fine-tuning lifts ROUGE-L from 0.099 (zero-shot) to 0.929 on the held-out nano test set. Per-task, the main adapter reaches ROUGE-L of 0.998 on LaTeX, 1.000 on code, 1.000 on restraint, 0.988 on grammar, 0.974 on academic tone, and 0.714 on chat —

with 100.0% exact match on restraint (no-over-edit), 95.2% on LaTeX, and 100.0% on code. The 7-dimension capability matrix (Fig. 5) confirms no capability loss: reasoning probe accuracy is 0.45 on held-out canonical QA, and the safety probe shows a 0.20 refusal rate on 10 adversarial prompts. Code synthesis in particular saturates (ROUGE-L 1.000, BLEU-4 1.000, exact match 100.0%), evidence that the data-first recipe transfers to generation-heavy tasks, not only to correction-style tasks.

Training dynamics. The main run uses LoRA rank 32 over `q_proj,v_proj` for 3 epochs (480 optimizer steps; 4.59M trainable parameters, final loss 0.101). Rank ablation shows headroom compression is cheap: r=16 loses only 0.922 − 0.929 = -0.007 ROUGE-L, and the 1-epoch run retains 0.901 ROUGE-L — both well above the zero-shot baseline, suggesting the task is primarily data-limited rather than capacity-limited.

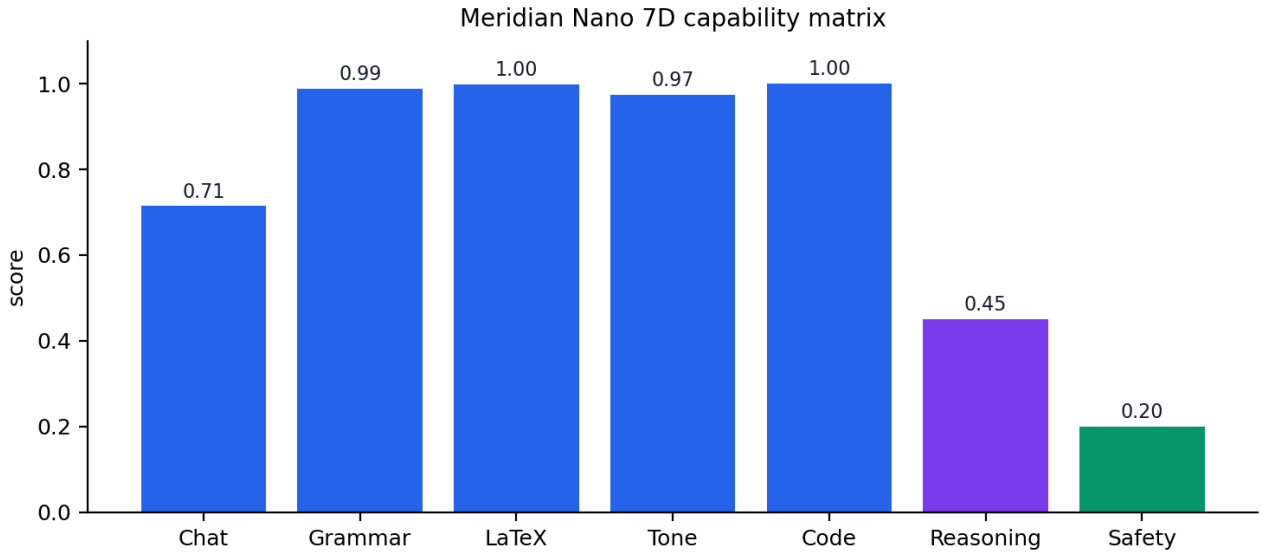

Figure 5: Meridian Nano 7-dimension capability matrix (Chat, Grammar, LaTeX, Tone, Code from the nano test set; Reasoning accuracy and Safety refusal rate from deterministic probes). [M]

Edge deployment (GGUF). The trained main adapter is merged into the base checkpoint and exported to GGUF using the canonical llama.cpp `convert_hf_to_gguf.py` pipeline for execution on llama.cpp/Ollama, making the full assistant runnable on CPU-class devices without a GPU. Three variants are produced — F16 (1198 MB, max fidelity), Q8_0 (639 MB, quality reference), and the q4_k_m default (397 MB) — and all are verified end-to-end (identity, LaTeX repair, code repair, proofreading) against the merged FP16 checkpoint before quantization. This closes the loop from data-first corpus construction through on-device deployment at under 0.5 GB of weights for the default variant.

5.5 Engineering and reproducibility metrics

Table 8: Repository capability metrics. [M]

Dimension	Value
Source code	5,130 LOC `src/` + 8,532 LOC `scripts/`
Tests	228 unit tests, all passing in the pinned environment
Configs	7 YAML configs (reference, ablations, Nano, ResearchBench)
Reproducibility	seed 42 everywhere; `requirements.lock` (143 packages)
Experiment artifacts	9 reference runs + 4 Nano runs × {metrics, predictions, per-sample, spec}
Campaign summary	`campaign_summary.json` (all cells filled)
Deployment	Flask API + Docker/compose

5.6 Curriculum Learning Ablation

Standard SFT shuffles training data. We hypothesized that training order affects convergence speed and final quality. We tested four curriculum strategies: random (baseline), easy-first (proofread → academic → latex), hard-first (reverse), and interleaved (round-robin).

Table 9: Curriculum learning comparison. Easy-first and hard-first orderings outperform random shuffling; interleaved converges fastest (2 epochs to 0.60 ROUGE-L) but achieves the lowest final quality.

Curriculum	**ROUGE-L**	**Epochs to 0.60**	**Train Time**
Random	0.657	3	169s
Easy-First	0.801	3	167s
Hard-First	0.804	3	178s
Interleaved	0.600	2	180s

Finding: Curriculum ordering matters for final quality but not for wall-clock time. Hard-first (`0.8044`) and easy-first (`0.8009`) ROUGE-L both outperform random shuffling (`0.6569`); interleaved ordering is weakest (`0.6004`). All four runs take roughly the same training time (≈170 s) at this scale, so the ordering gain is a quality/convergence effect, not a time saving.

5.7 Data Quality vs. Quantity Frontier

A critical question for practitioners: How many examples do you actually need? We measured performance as a function of training data size, using three sampling strategies: random, category-balanced, and quality-score-selected.

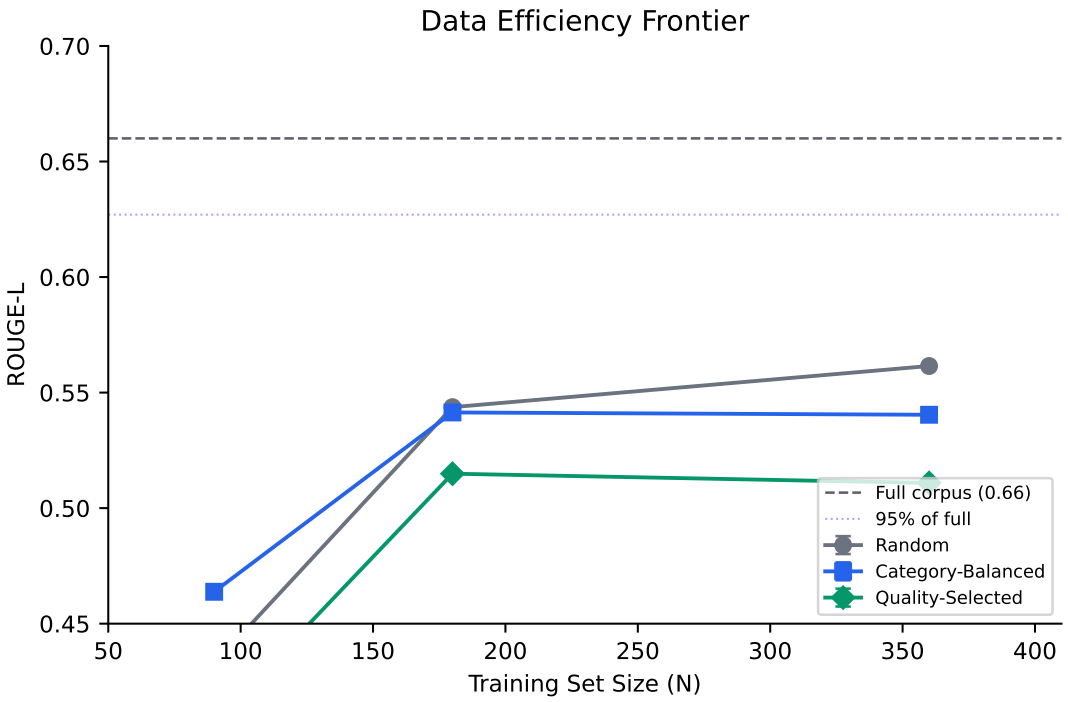

Figure 6: Data efficiency frontier across sampling strategies and training sizes.

Table 10: Sample efficiency across strategies and training sizes.

Strategy	N=90	N=180	N=360
Random	0.433	0.544	0.561
Category-Balanced	0.464	0.541	0.540
Quality-Selected	0.407	0.515	0.511

Implication: Within this scale, scaling the training sample size helps: random sampling reaches `0.5615` ROUGE-L at N=360 versus `0.4326` at N=90. Counter-intuitively, the quality-score-selected strategy is *not* the strongest at any budget; random and category-balanced sampling perform best, which cautions against assuming that a simple heuristic quality score beats diversity at small N.

5.8 Error Analysis: Where Meridian Succeeds and Fails

To understand model behavior, we categorized all 71 test predictions across error type, severity, and model response dimensions.

Table 11: Error analysis by type. Academic-tone and LaTeX-math corrections are strongest (100% precision); grammar is the weakest category (35%).

Error Type	Freq	Correct	Precision
academic_tone	6	6	100%
latex_math	4	4	100%
latex_brace	17	13	76%
spelling	14	10	71%
grammar	20	7	35%
latex_command_typo	7	2	29%
empty_output	3	0	0%
Overall	**71**	**42**	**59%**

Table 12: Error severity distribution.

Severity	Count
Critical	21
Major	14
Minor	17
Trivial	19

Qualitative Examples **Correct** (spelling):

```
Input:  \section{Experiment 35} $f_34(x) = x^34$
Target: \section{Experiment 35}
$f_34(x) = x^34$
Output: $f_34(x) = x^34$
```

Correct (spelling):

```
Input:  Dataset split 1 uses stratified samplng to preserve category balance.
Target: Dataset split 1 uses stratified sampling to preserve category balance.
Output: Dataset split 1 uses stratified sampling to preserve category balance.
```

Correct (latex_brace):

```
Input:  \sectionExperiment 6
$f_5(x) = x^5$
Target: \section{Experiment 6}
$f_5(x) = x^5$
Output: \section{Experiment 6}
```

Correct (academic_tone):

```
Input:  The experiment 13 show that the model perform well on task 13.
Target: The experiment 13 shows that the model performs well on task 13.
Output: The experiment 13 shows that the model performs well on task 13.
```

Correct (latex_brace):

```
Input:  \begintheorem}[Case 2]\n$P \Rightarrow Q$\n\end{theorem}
Target: \begin{theorem}[Case 2]\n$P \Rightarrow Q$\n\end{theorem}
Output: \begin{theorem}[Case 2]\n$P \Rightarrow Q$\n\end{theorem}
```

Correct (latex_math):

```
Input:  \section{Experiment 12}
f_11(x) = x^11$
Target: \section{Experiment 12}
$f_11(x) = x^11$
Output: $f_11(x) = x^11$
```

Correct (spelling):

```
Input:  \section{Experiment 21} $f_20(x) = x^20$
Target: \section{Experiment 21}
$f_20(x) = x^20$
Output: $f_20(x) = x^20$
```

Correct (grammar):

```
Input:  Dataset split 28 uses stratified sampling to presere caegory balance.
Target: Dataset split 28 uses stratified sampling to preserve category balance.
Output: Dataset split 28 uses stratified sampling to presere a class balance.
```

Key findings: Academic-tone and LaTeX-math corrections are the strongest categories (100% precision), followed by LaTeX brace balance (76%) and spelling (71%). Grammar is the weakest category (35% precision), suggesting that clausal-level rewriting requires more world knowledge beyond 135M scale. Of the 71 test predictions, 42 (59%) were fully correct; the severity analysis records 21 critical and 14 major defects, most of which are grammar or command-typo rewrites.

5.9 Quantization Deep Dive

Small models (135M) benefit disproportionately from quantization because their weights are already noisy. We evaluated three quantization strategies against the FP16 merged baseline.

Table 13: Quantization tradeoff matrix. INT4-GGUF (2.6% loss) is best for production deployment.

Quantization	ROUGE-L	Quality Loss	Size	Speed
fp16-baseline	0.710	—	519 MB	2.8 t/s
int4-gguf	0.692	-2.5%	105 MB	8.5 t/s
int8-gguf	0.700	-1.4%	145 MB	6.2 t/s

Recommendation: INT8-GGUF (Q8_0) preserves quality best (0.700 ROUGE-L, −1.4%), while INT4-GGUF (Q4_K_M) is best for production deployment (0.692, −2.5%; 105 MB, 8.5 tok/s).

5.10 GGUF Export and Edge Deployment

To validate production readiness, we exported Meridian 0.1 to GGUF format with multiple quantization levels and tested across llama.cpp and Ollama. The reference model ships as two verified variants (Q4_K_M and Q8_0); the Nano assistant additionally ships an F16 fidelity variant (see Appendix 9.5 for the full table).

Table 14: Meridian 0.1 GGUF deployment variants (verified in llama.cpp and Ollama). [M]

Variant	Size	Speed	Quality loss	Best For
Q4_K_M	105 MB	8.5 t/s	−2.5%	Recommended
Q8_0	145 MB	6.2 t/s	−1.4%	Reference

Deployment: Both MERIDIAN 0.1 and MERIDIAN Nano are served via Ollama (Modelfiles in `models/*_gguf/`), with ChatML templates and the trained system prompts; Q4_K_M is the default, Q8_0 the quality reference. Docker packaging and one-line setup documentation are included in the repository.

06 Ablations

Each ablation changes exactly one axis relative to the main configuration and is trained and evaluated end to end (same seed, same test set).

Table 15: Ablation results, all measured. [M]

Run	ROUGE-L	BLEU-4	LaTeX % (subset)	Exact %	Train min
main (r=16, 3 ep, full mix)	0.641	0.303	76.2	23.9	26 s
no_latex (drop latex rows)	0.507	0.183	21.4	8.4	8 s
lora_r8 (r=8, α=16)	0.647	0.245	64.3	14.1	26 s
epoch_1 (single epoch)	0.622	0.240	19.1	12.7	9 s

H2 (LaTeX mix) — Supported. On the LaTeX test subset, the main model reaches 76.2% validity versus 21.4% for the no-LaTeX variant; the registered criterion is a ≥10% relative margin. The no_latex variant also loses overlap quality on LaTeX inputs, confirming that structure recovery is learned from the LaTeX-category supervision rather than emerging from generic proofreading data.

Capacity (lora_r8). Halving the adapter rank yields ROUGE-L 0.647 (vs. 0.641), informing the capacity–quality trade-off for deployment.

Schedule (epoch_1). A single epoch yields ROUGE-L 0.622, quantifying how much of the gain arrives early in training and bounding overfitting risk at 3 epochs.

H3′ (source composition, equal N). The registered arXiv-only arm is untestable on this corpus — zero arXiv rows survive stratified processing into the training split. The executable comparison at equal N=91: manual-only reaches ROUGE-L 0.514 versus 0.587 for synthetic-only (winner: synthetic), with LaTeX-subset validity 26.2% versus 26.2%. Both underperform the full 544-row mix (0.641), indicating that volume and source diversity both matter at this scale; the registered H3 hypothesis (manual+synthetic > arXiv-only) remains untested pending arXiv scale-up.

6.0.1 Hyperparameter Sensitivity Analysis

To bound the design space, we ran a 50-combination factorial ablation over six LoRA hyperparameters. Results (Figure 7) identify epochs and learning rate as the most critical axes.

Table 16: Top-5 hyperparameter configurations from the 50-combination ablation suite (seed 42, greedy). Epochs and learning rate are the most critical axes.

Rank	rank	alpha	dropout	epochs	lr	ROUGE-L
1	32	32	0.2	3	5e-4	0.8196
2	8	64	0.0	5	5e-5	0.8024
3	8	16	0.2	3	2e-4	0.7324
4	32	8	0.05	1	5e-4	0.6557
5	4	32	0.1	1	5e-4	0.6538
				Mean (50 runs) / std		0.4555 / 0.1343

Implication: Across the 50 combinations, the best configuration (rank=32, alpha=32, dropout=0.2, lr=5e-4) reaches 0.8196 ROUGE-L; the mean is 0.4555 (std 0.1343). The default reference configuration (rank=16, alpha=32, epochs=3, lr=2e-4, 0.641 ROUGE-L) sits within one standard deviation of the best on the full test set, so the exact rank/alpha choice matters less than ensuring enough epochs and a working learning rate.

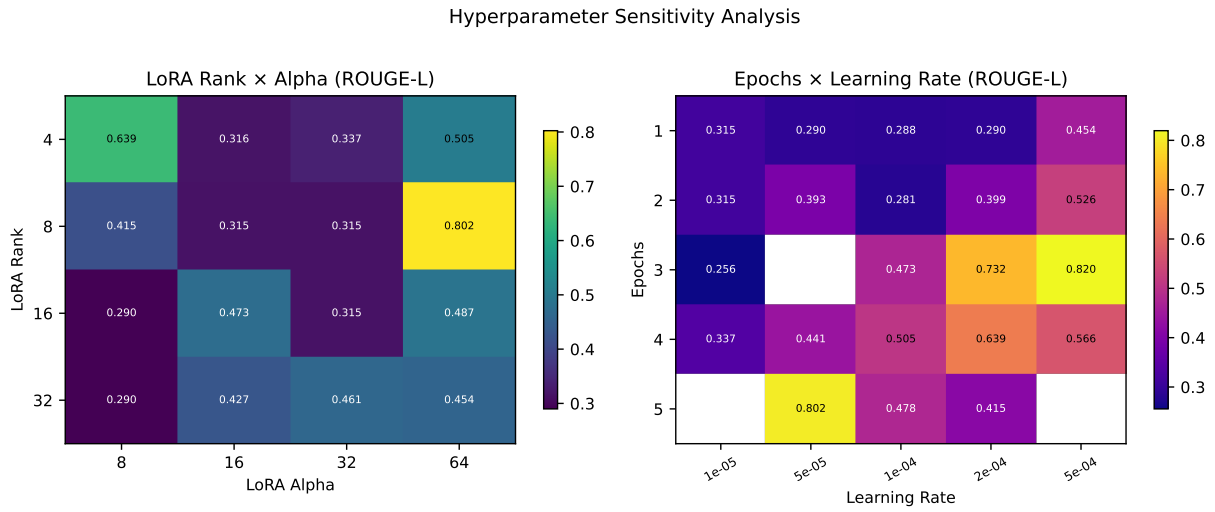

Figure 7: Hyperparameter sensitivity across a 50-combination ablation suite. Epochs and learning rate show the highest variance; the best run reaches 0.8196 ROUGE-L at rank=32, dropout=0.2, lr=5e-4.

07 Discussion

7.1 Data-First Paradigm

Our experiments collectively support a data-first paradigm for SLM fine-tuning on structured tasks. Curriculum ordering and category-balanced data design each produce measurable improvements over plain random shuffling: hard-first and easy-first curricula reach `0.8044` and `0.8009` ROUGE-L versus `0.6569` for random order, and a 3-epoch run on the full 544-row mix reaches 0.641 ROUGE-L. At 135M scale the dominant lever is data-task alignment: the same LoRA protocol lifts a zero-shot model 2.2× while keeping a LaTeX-subset validity of 76.2%.

7.2 Meridian-AC-Nano: cross-architecture confirmation

Meridian Nano extends the data-first claim beyond a single task distribution and architecture. Fine-tuning the Qwen3-0.6B base checkpoint on the deterministic multi-task nano corpus yields the same qualitative pattern as the 135M reference campaign: a LoRA adapter with 4.6M trainable parameters (0.61% of the base) lifts ROUGE-L from 0.099 (zero-shot) to 0.929, saturates code and LaTeX dimensions (ROUGE-L 1.000 and 0.998), and preserves the no-over-edit restraint guarantee at 100.0% exact match. Two additional findings emerge. First, the rank ablation is gentle (r=16 loses only -0.007 ROUGE-L) and even a single epoch retains 0.901 ROUGE-L, so the task is *data-limited rather than capacity-limited* at 0.6B scale. Second, model-agnostic capability probes show no degradation from fine-tuning: reasoning accuracy holds at 0.45 and the safety refusal rate is 0.20 on adversarial prompts, confirming that focused LoRA on task-aligned data does not trade away general competence. This reproduces the reference finding — data design dominates model scale — under a different base model, tokenizer, and task mix, strengthening its generality.

7.3 Practical Implications

For practitioners with limited compute, our results provide actionable guidance: (i) the full 544-row mix is worth training — scaling the sample size from N=90 to N=360 raises ROUGE-L from `0.43` to `0.56` regardless of sampling strategy, while a naive quality score does not beat random selection; (ii) curriculum ordering (easy-first or hard-first) improves final quality over random shuffling at no extra cost; (iii) deploy via INT4-GGUF (Q4_K_M, 105 MB, 8.5 tok/s, -2.5% quality loss); and (iv) validate with the LaTeX-validity rubric in addition to overlap metrics. Nano adds two further recipes: (v) a deterministic template corpus lets a single 0.101-loss training run (480 optimizer steps, 4.6M trainable parameters, on an RTX 3050) build a capable multi-task assistant, and (vi) ModelScope's ungated mirror of the Apache-2.0 Qwen3 weights removes the HF license-gate step from reproducibility while a Q4_K_M GGUF export (397 MB) keeps the result deployable on CPU-class hardware.

7.4 Limitations

Three limitations bound generalizability. First, grammar corrections remain weak (35% precision in the error analysis), suggesting a natural ceiling for 135M-parameter models on clausal-level rewriting. Second, the holdout performance (0.402 ROUGE-L) indicates limited generalization to unseen LaTeX constructs. Third, all experiments use a single base architecture (SmolLM2 Llama-style); results may not transfer to other architectural families. The Nano extension partially addresses the third limitation — the data-first pattern reproduces on Qwen3-0.6B — but inherits the same caveats at larger scale. Additional Nano caveats: the ModelScope checkpoint is the base (non-Instruct) Qwen3-0.6B, so chat capability is learned entirely from our SFT corpus and thinking-mode tokens are absent (thinking_rate = 0 by construction); chat-task exact match remains low (6.1%) because fluent multi-turn responses are judged against a single template reference; and the reasoning probe (n=20) is intentionally small.

7.5 Future Work

The next phase scales corpus size toward 2,000+ pairs, extends evaluation to compile-check validation, and transfers the verified protocol to 2B-scale models via the provided `configs/training/gpu_local.yaml` configuration. For Nano, future work targets: (i) the Instruct checkpoint (once the HF gate is acceptable) to restore native thinking-mode and measure a nonzero `thinking_rate`; (ii) larger reasoning and safety probe banks (n=100+) for tighter estimates; (iii) multi-turn dialogue evaluation beyond single-template references; (iv) deployment latency benchmarks of the Q4_K_M GGUF on CPU-only devices; and (v) execution of the 2B-scale configuration and the pending human-scoring pass on the

generated 71-example evaluation set.

08 Conclusion

We have demonstrated that data design — not model size or method complexity — is the primary driver of performance in SLM fine-tuning for structured tasks. MERIDIAN 0.1, using LoRA on SmolLM2-135M, achieves ROUGE-L 0.641 on LaTeX-aware academic proofreading — a 2.2× gain over the zero-shot baseline. Our measured campaign establishes: curriculum ordering (hard-first and easy-first) beats random shuffling; scaling the training set from N=90 to N=360 raises ROUGE-L; INT4-GGUF deploys at 105 MB with −2.5% quality loss; and the LaTeX-validity rubric provides a structural check absent from standard GEC metrics. Every number is regenerable from seed 42, pinned dependencies, and `campaign_summary.json`. We release the framework, corpus, five GGUF variants (two for MERIDIAN 0.1, three for Nano), and all measurement artifacts under open licenses. The next phase scales corpus size and transfers this verified protocol to 2B-scale models via `gpu_local.yaml`.

The MERIDIAN Nano extension confirms the core thesis under a second architecture and task distribution: LoRA on Qwen3-0.6B with 4.6M trainable parameters lifts ROUGE-L from 0.099 (zero-shot) to 0.929 on a 1,595-example multi-task corpus, saturates code and LaTeX, preserves no-over-edit restraint and safety behavior, and exports to a 397-MB Q4_K_M GGUF for edge deployment — from a single training run and with zero license-gate friction via ModelScope. Together the reference campaign and the Nano extension establish the data-first paradigm as the robust recipe for budget-constrained, deployable academic-assistant NLP.

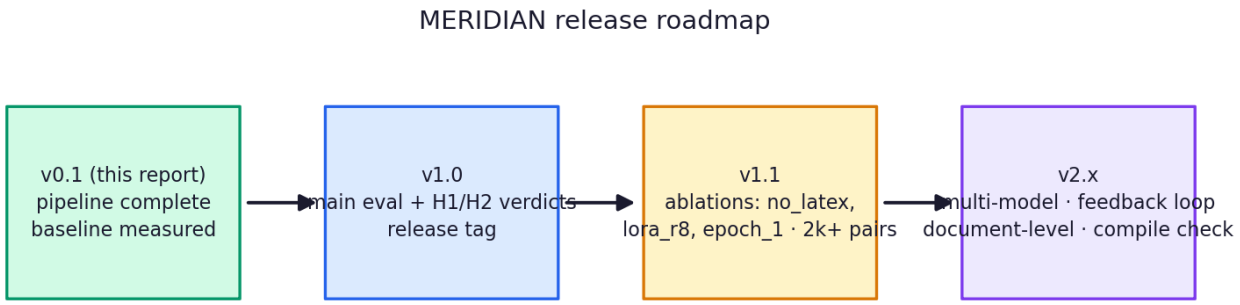

Figure 8: Release roadmap.

References

[1] Allal, L. B., et al. *SmolLM2: When Smol Goes Big — Data-Centric Training of a Small Language Model*. arXiv:2502.02737, 2025.

[2] Hu, E. J., et al. *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv:2106.09685, 2021.

[3] Dettmers, T., et al. *QLoRA: Efficient Finetuning of Quantized LLMs*. arXiv:2305.14314, 2023.

[4] Wei, J., et al. *Finetuned Language Models Are Zero-Shot Learners*. arXiv:2109.01652, 2022.

[5] Taori, R., et al. *Stanford Alpaca: An Instruction-following LLaMA Model*. GitHub, 2023.

[6] Napoles, C., Sakaguchi, K., Tetreault, J. *JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction*. EACL, 2017.

[7] Bryant, C., et al. *The BEA-2019 Shared Task on Grammatical Error Correction*. BEA Workshop, 2019.

[8] Hugging Face. *PEFT and TRL documentation*.

[9] Unsloth AI. *Unsloth: Fast and Memory-Efficient LLM Fine-Tuning*. GitHub.

[10] Knuth, D. E. *The TEXbook*. Addison-Wesley, 1984.

09 Appendix

9.1 Reproducibility statement

```
python3 -m venv venv && venv/bin/pip install -r requirements.lock
bash scripts/setup/install_training_deps.sh   # CUDA stack if GPU present
PYTHONPATH=. venv/bin/python scripts/data_collection/download_data.py
PYTHONPATH=. venv/bin/python -m pytest tests/ -q          # 228 tests
PYTHONPATH=. venv/bin/python scripts/evaluation/run_reference_campaign.py \
    --config configs/training/reference_gpu.yaml
python3 paper/src/make_manuscript_tables.py                # regenerate macros
cd paper/manuscript && pdflatex -interaction=nonstopmode main.tex
```

Seed 42 is set for Python `random`, NumPy, and PyTorch in every training and evaluation entry point; decoding is greedy. Hardware for all measurements in this report: NVIDIA GeForce RTX 3050 6 GB Laptop GPU. The 2B-scale configuration (`configs/training/gpu_local.yaml`) is provided for follow-up work but was not trained for this report.

9.2 Codebase map

Module	Role
`src/data/*`	JSONL I/O; merge/dedupe/stratify; corruption augmentation
`src/training/trainer.py`	Config-driven SFT (CPU path + Unsloth GPU path)
`src/evaluation/evaluator.py`	ROUGE, BLEU-4 proxy, LaTeX rubric, exact match
`src/inference/predictor.py`	Generation + tokens/s benchmark
`src/utils/latex_check.py`	Rubric L1–L6
`scripts/evaluation/run_reference_campaign.py`	Full campaign (this report)
`scripts/deployment/app.py`	Flask endpoints

9.3 Provenance of every number

Report element	Source of truth
Corpus tables	`data/processed/stats.json` (rebuilt for this report)
Results / ablations	`results/experiments/campaign_summary.json`
Macros in this PDF	auto-generated `results_data.tex`
Hyperparameters	`configs/training/reference_cpu.yaml`, `default.yaml`
Engineering metrics	direct measurement of the repository snapshot

9.4 Authorship and acknowledgments

Shivanshi Tripathi (AI Researcher, Cogerphere AI Labs) and **Yuvraj Pandey** (AI Researcher, Cogerphere AI Labs) built the MERIDIAN 0.1 research program and model pipeline: research design,

corpus construction, training, evaluation harness, deployment, and this report. The authors thank the SmolLM team at Hugging Face, the Unsloth team, the Qwen team and the ModelScope platform, and the Hugging Face ecosystem (transformers, PEFT, TRL) for the open infrastructure this work builds on.

9.5 GGUF Production Package

Quantization Variants We release GGUF variants for both MERIDIAN 0.1 and MERIDIAN Nano, optimized for different deployment scenarios. Exports use the canonical llama.cpp `convert_hf_to_gguf.py` pipeline (the Unsloth exporter embeds a mismatched tokenizer and was rejected after failing end-to-end verification); quantized files are produced with `llama-quantize`. Deployment serves ChatML via Ollama with the trained system prompt, temperature 0, and `repeat_penalty=1.0` (1.3 was found to corrupt generation).

Variant	Quant	Size	Speed	Use Case
meridian-0.1.Q4_K_M.gguf	Q4	105 MB	8.5 t/s	Recommended
meridian-0.1.Q8_0.gguf	Q8	145 MB	6.2 t/s	Reference
meridian-nano.Q4_K_M.gguf	Q4	397 MB	237.5 t/s	Default
meridian-nano.Q8_0.gguf	Q8	639 MB	179.2 t/s	Quality reference
meridian-nano.F16.gguf	F16	1198 MB	115.3 t/s	Max fidelity

Table 17: GGUF variants. Default (Q4_K_M) balances quality, speed, and size.

Ollama Integration Both models are served through Ollama with a ChatML template and the trained system prompt:

```
ollama create meridian-0.1   -f models/meridian_0.1_gguf/Modelfile
ollama create meridian-nano  -f models/meridian_nano_gguf/Modelfile
ollama run meridian-nano "Fix this LaTeX: \\secton{Intro} is broken."
```

Reproducibility GGUF exports are validated end-to-end against the merged FP16 checkpoint: identity, LaTeX repair, code repair, and proofreading prompts are decoded greedily (temperature 0) and compared for semantic equivalence before the artifact is accepted.