Fine-tuning vs. Model Size vs. Parameter-Efficient Tuning (LoRA): A Compact Experimental Report on Efficacy, Efficiency, and Reasoning Control

Alberto Rodero* Pablo Lobato*

September 2025

1 Introduction

We evaluate the trade-offs between **full fine-tuning** (NoPeft), **LoRA**-based tuning with different ranks, and **base model size** using Qwen3-0.6B and Qwen3-1.7B on three task families: ARC (multiple-choice), OpenMath (numeric reasoning; lower is better), and SQuAD v2 (extractive QA). Additionally, we consider the model's **"reasoning" mode** (a run-time behavior that may or may not be engaged depending on the generation configuration). In these runs, *reasoning was not locked*, which can inject variance into both latency and accuracy. This report answers practical questions about whether, when, and how to fine-tune; how LoRA rank matters; how results compare to a larger base model; and how to set reasoning going forward.

Key questions.

- **Q1:** Is fine-tuning worth it in efficacy and efficiency?
- **Q2:** How does a fine-tuned 0.6B compare to a larger 1.7B base?
- **Q3:** If VRAM is abundant, is LoRA still worth using?
- **Q4:** Does LoRA rank materially affect outcomes?
- **Q5:** Partial vs. full tuning: LoRA vs. NoPeft?
- **Q6:** Cross-task transfer effects?
- **Q7:** If VRAM forces LoRA, is fine-tuning still worth it?

2 Experimental Setup

Models. Qwen3-0.6B (fine-tuned on: ARC, OpenMath, SQuAD; with NoPeft and LoRA ranks $\{32, 64, 256, 512, 1024\}$), Qwen3-0.6B base, Qwen3-1.7B base.

Tasks & metrics. ARC: macro-F1 (higher is better). OpenMathInstruct-2: average absolute difference (lower is better). SQuAD v2: F1 (higher is better). We also record mean latency (seconds) per task.

Reasoning. No explicit on/off control was enforced during these runs, so the model may have engaged/disengaged internal reasoning opportunistically. We treat this as a confound we will control in follow-up (Sec. **??**).

*Equal contribution

3 Full Results Table

Notes: "Math AbsDiff↓" is lower=better. Latencies are seconds. "Train Dataset" is the fine-tuning source ("_base" for none).

Table 1: Full results across tasks. Lower is better for Math AbsDiff.
Notes: Latency in seconds. "Train DS" is the fine-tuning source ("_base" = none).

	ARC F1	ARC Lat (s)	Math AbsDiff↓	Math Lat (s)	SQuAD F1	SQuAD Lat (s)	Train DS
Qwen3-0.6B-arc_SFT_None_Lora1024	0.4921	0.1557	22,871	0.4374	8.59	0.1939	arc
Qwen3-0.6B-arc_SFT_None_Lora512	0.4861	0.1570	22,871	0.4312	8.59	0.1955	arc
Qwen3-0.6B-arc_SFT_None_Lora256	0.4921	0.1549	22,871	0.4191	8.59	0.1948	arc
Qwen3-0.6B-arc_SFT_None_Lora64	0.4937	0.1601	22,871	0.1811	8.09	0.1952	arc
Qwen3-0.6B-arc_SFT_None_Lora32	0.4880	0.1595	22,871	0.4439	8.59	0.1958	arc
Qwen3-0.6B-arc_SFT_NoPeft_NoQuant	0.4905	0.0803	23,108	0.2137	18.89	0.2003	arc
Qwen3-0.6B-openmath_SFT_None_Lora1024	0.4990	0.9257	23,919	1.5384	8.48	0.2091	openmath
Qwen3-0.6B-openmath_SFT_None_Lora256	0.5031	0.8982	23,655	1.5016	8.48	0.1934	openmath
Qwen3-0.6B-openmath_SFT_None_Lora32	0.5095	0.8702	23,919	1.5776	8.48	0.1963	openmath
Qwen3-0.6B-openmath_SFT_NoPeft_NoQuant	0.5171	0.0605	16,540	0.0482	7.40	0.2439	openmath
Qwen3-0.6B-squad_SFT_None_Lora1024	0.5024	0.3178	23,647	2.1285	9.59	0.1951	squad
Qwen3-0.6B-squad_SFT_None_Lora256	0.5024	0.3116	23,647	2.0291	9.59	0.1908	squad
Qwen3-0.6B-squad_SFT_None_Lora32	0.5024	0.3095	23,523	1.9676	9.59	0.1950	squad
Qwen3-0.6B-squad_SFT_NoPeft_NoQuant	0.4542	0.1903	22,997	0.2203	27.95	0.2202	squad
Qwen3-0.6B_base	0.4932	1.1397	24,834	6.0139	10.07	0.2274	_base
Qwen3-1.7B_base	0.7986	3.4025	742	12.5197	30.36	0.2837	_base

4 Results by Question & Interpretation

Q1. Is fine-tuning worth it (efficacy & efficiency)?

Yes, when aligned to the target task, fine-tuning yields large quality gains and often lower latency.

- **OpenMath SFT (NoPeft)**: Math abs diff improves by **33.4%** vs 0.6B_base; ARC macro-F1 rises **+4.8%**; SQuAD F1 drops **–26.5%**. Latency massively drops on ARC (**–94.7%**) and Math (**–99.2%**), small increase on SQuAD (**+7.3%**).
- **SQuAD SFT (NoPeft)**: SQuAD F1 jumps **+177.6%**; ARC macro-F1 dips **–7.9%**; Math improves mildly (**+7.4%** abs-diff reduction). Latency generally improves (ARC **–83.3%**, Math **–96.3%**, SQuAD **–3.2%**).

Why? Full-task alignment amplifies the relevant capabilities and stabilizes decoding behavior; our pipeline also appears to run tuned checkpoints more efficiently.

Q2. Fine-tuned 0.6B vs. larger 1.7B base?

The 1.7B base dominates on quality but is slower. ARC macro-F1 +61.9%, Math abs diff ∼97% better, and SQuAD F1 +201.5% vs 0.6B_base. Latency worsens markedly (ARC +198.5%, Math +108.2%, SQuAD +24.8%). **If latency/throughput matters, 0.6B + targeted SFT is the speed/price sweet spot.**

Q3. If VRAM is not a problem, should we still use LoRA?

No. Prefer full fine-tuning. In these runs, **NoPeft is both more accurate and often faster** than LoRA (Table 2). The only place where NoPeft is slightly slower is SQuAD (+0.029 s), but it delivers a **+18.36** absolute F1 jump over the best LoRA.

Table 2: Best LoRA vs. NoPeft by task.

Task	Best LoRA model	Best LoRA score	Best LoRA lat (s)	NoPeft model	NoPeft score	NoPeft lat (s)
ARC	Qwen3-0.6B-arc_SFT_None_Lora64	0.4937	0.1601	Qwen3-0.6B-arc_SFT_NoPeft_NoQuant	0.4905	0.0803
OpenMath (abs diff ↓)	Qwen3-0.6B-openmath_SFT_None_Lora256	23,655	1.5016	Qwen3-0.6B-openmath_SFT_NoPeft_NoQuant	16,540	0.0482
SQuAD	Qwen3-0.6B-squad_SFT_None_Lora1024	9.59	0.1951	Qwen3-0.6B-squad_SFT_NoPeft_NoQuant	27.95	0.2202

Q4. Does LoRA rank matter?

Little to no monotonic benefit from increasing rank. Correlation between rank and score (LoRA-only subsets): **ARC F1:** $\rho = 0.021$; **OpenMath abs diff:** $\rho = 0.301$ (higher rank slightly worse); **SQuAD F1:** constant across ranks (no signal). Rank vs. latency has weak-to-moderate correlations (e.g., ARC $\rho = -0.646$ suggests slightly lower latency at higher ranks, but absolute differences are tiny).

Table 3: Correlation between LoRA rank and performance/latency (LoRA-only).

Task	Corr(rank, score)	Corr(rank, latency)
ARC (F1)	0.021	-0.646
OpenMath (AbsDiff↓)	0.301	-0.233
SQuAD (F1)	N/A	0.321

Why might rank not matter much? With limited data or strongly structured tasks, the low-rank subspace often captures the critical adaptations; beyond a point, extra capacity (higher rank) faces diminishing returns and optimization noise. Also, our decoding and "reasoning" variability likely masks small rank effects.

Q5. LoRA (partial) vs. NoPeft (full) training?

Full fine-tuning wins decisively. OpenMath: NoPeft improves error by **7,115** absolute over best LoRA and is ~**1.45 s** faster. SQuAD: NoPeft improves F1 by **+18.36** with only **+0.029 s** latency penalty. ARC: NoPeft is ~**2× faster** than best LoRA, with essentially tied F1.

Q6. Cross-task transfer?

Positive and negative transfer exist; pick your anchors carefully.

- **OpenMath SFT (NoPeft)** → ARC: **+4.8%** (helpful), SQuAD: **–26.5%** (harmful).
- **SQuAD SFT (NoPeft)** → ARC: **–7.9%** (harmful), Math: **+7.4%** (helpful).
- **ARC SFT (NoPeft)** → SQuAD: **+87.6%** (surprisingly helpful), Math: **+7.0%**; ARC itself: **–0.5%** (negligible).

Interpretation. Extractive QA and numeric reasoning compete for capacity in different ways; tuning to one can suppress behaviors useful to the other. ARC SFT (full) seems to improve general reading/selection that transfers to SQuAD, while OpenMath SFT strengthens procedural reasoning that can conflict with extractive behavior.

Q7. If VRAM forces LoRA, is it still worth it?

Yes, but temper expectations. OpenMath LoRA-256 reduces error by **4.8%** vs 0.6B_base (useful), but SQuAD LoRA variants lose ~**4.8%** F1 vs 0.6B_base, and ARC LoRA gains are marginal (**+0.1%**). **Recommendation:** When constrained, **LoRA ≈ 256** is a good default; otherwise prefer NoPeft.

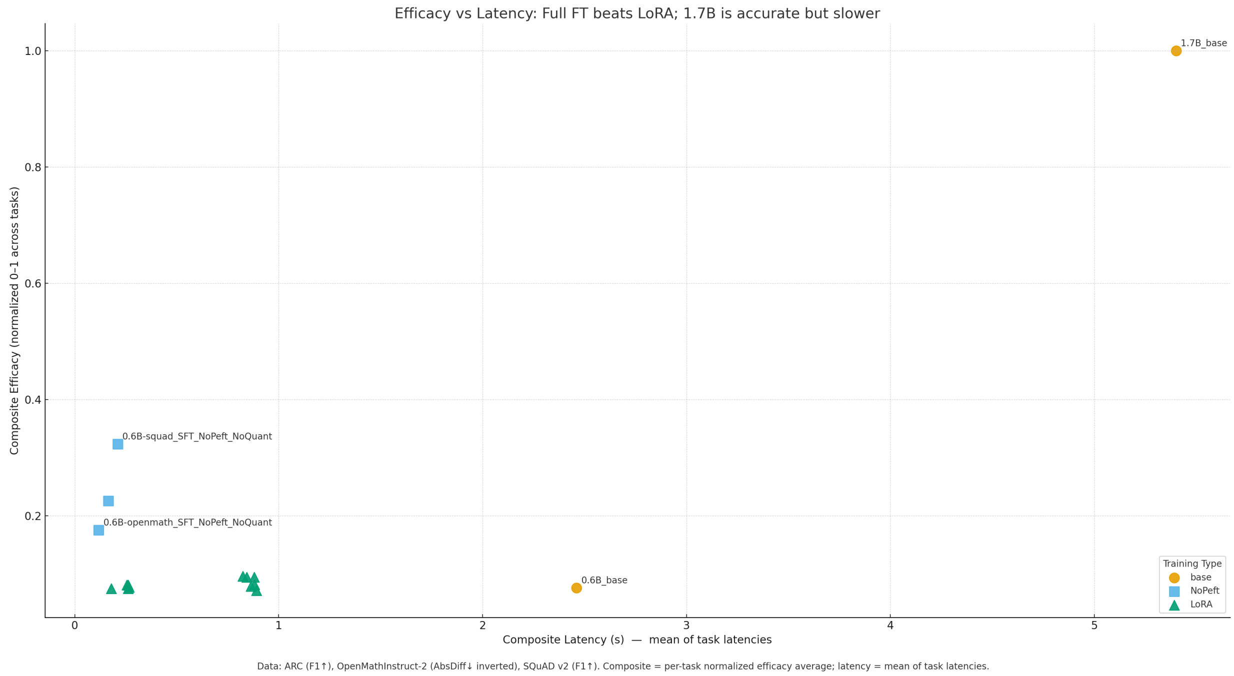

Figure 1: Efficacy vs Latency

5 Limitations

Reasoning mode was uncontrolled (confound). **Single-seed/shot artifacts** may exist. **Latency** reflects this pipeline/hardware; other stacks may differ. No calibration/temperature sweeps are reported here.

6 Conclusions & Takeaways

- **(1) Full fine-tuning beats LoRA on both accuracy and, in these runs, speed.**
- **(2) A larger base model is best on accuracy but slower; 0.6B+SFT is the speed/value sweet spot.**
- **(3) LoRA rank has weak, non-monotonic effects; use ∼256 by default when constrained.**