Punching Above Their Weight: Classification-Head Fine-Tuning of Tiny Language Models (TLMs) for Verifiable Multiple-Choice Tasks

Anonymous ACL submission

Abstract

We define *Tiny Language Models* (TLMs) as models below roughly 3B parameters that fit on mainstream consumer devices. We study how to adapt them for and use them on verifiable multiple-choice tasks. We compare three LoRA-based fine-tuning paradigms (label generation, gold only, and our discriminative classification head) on a unified setup across several Qwen3 models from 0.6B to 8B and five benchmarks: HellaSwag, WinoGrande, PIQA, SciQ and ARC-C. Classification-head fine-tuning reliably outperforms label generation (+2–3%) at the 0.6B and 1.7B scales. Further, TLMs fine-tuned using the discriminative method are competitive to zero-/few-shot GPT-3 (175B), PaLM (540B) and GPT-4. The performance we report for Qwen3-0.6B and Qwen3-1.7B are SOTA on HellaSwag, WinoGrande, and PIQA.

1 Introduction

Running large language models (LLMs) on common consumer hardware has become increasingly important as the capabilities and ubiquity of LLMs rise at unprecedented pace. Yet, vast majority of published research focuses on LLMs that cannot be effectively run on such devices. Hence, we argue that there is a great need to coin a novel notion of a Tiny LM (TLM) that is motivated by contemporary practical constraints stemming from common consumer hardware. Because TLMs remain understudied their deployments are often sub-optimal with lower accuracy than what they can deliver.

This study of fine-tuning paradigms and evaluation approaches for TLMs offers actionable guidance on how to adapt and use a TLM. The study is task-agnostic and only assumes a setting in which there is a finite set of candidate outputs for each data point with one of the candidates deemed as correct (e.g., classification, span annotation, or typical applications of LLM-as-a-judge). We denote such tasks as verifiable multiple-choice. Our specific focus is on commonsense reasoning tasks as these would likely be the main TLMs use case on local devices. However, such tasks have been shown to be challenging for LMs of small sizes (Li et al., 2025; Lu et al., 2024a).

This study makes the following **contributions**:

1. It defines and motivates the practical notion of *Tiny Language Models (TLMs)*.
2. It describes a fine-tuning approach based on discriminative classification head suitable for application of TLMs on multiple-choice tasks.
3. It is the first comparison of the three fine-tuning paradigms across five well-established benchmarks focused on TLMs, and the reported performance on most of them is SOTA.

The main **findings** of this study are:

a) In TLMs, the fine-tuning based on classification head offers significantly better performance than approaches based on fine-tuning for next-token (label) prediction.

b) Modifying seemingly mundane details of inference can result in dramatic discrepancies in observed performance of a TLM.

c) In larger models ($>$ 3B), the gap is closed and there is no significant difference between the fine-tuning approaches.

d) Additionally, the memorisation-intensive science benchmarks benefit less from the classification approach, but with no statistically significant loss either.

Our findings suggest that the previously reported performance of TLMs may be simply an artifact of how they are typically fine-tuned and utilized (often sub-optimally). This calls for careful rethinking of evaluation and progress in this space. While it

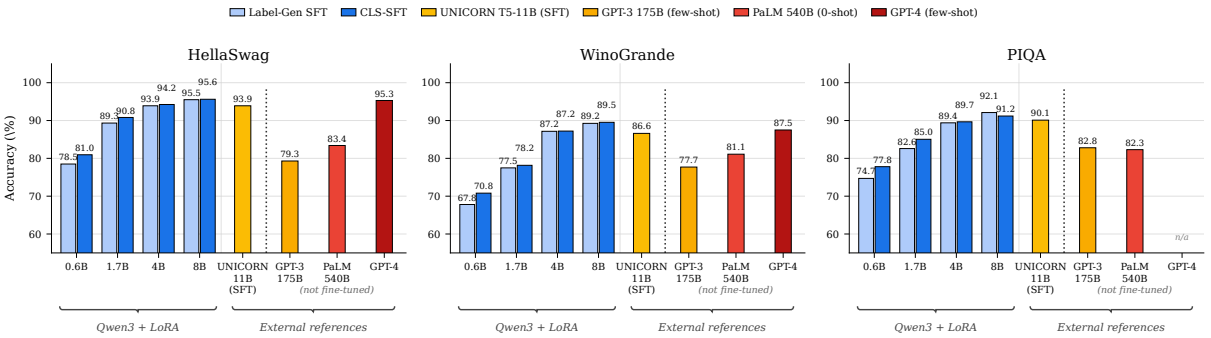

Figure 1: Accuracy on three commonsense benchmarks for Qwen3 + LoRA models fine-tuned with Label-Gen SFT and Classification-head SFT (CLS), compared to four external reference models: UNICORN T5-11B (Lourie et al., 2021), GPT-3 175B (Brown et al., 2020a), PaLM 540B (Chowdhery et al., 2023), and GPT-4 (OpenAI, 2023). UNICORN is the only fine-tuned external reference. PIQA is not reported in the GPT-4 Technical Report. We observe that at the 1.7B scale, fine-tuned Qwen3 already exceeds the reported GPT-3 and PaLM 540B results on HellaSwag and PIQA, and exceeds GPT-3 on WinoGrande, despite being roughly 100× smaller than GPT-3 and 320× smaller than PaLM.

is generally accepted that TLMs do not perform adequately on common reasoning tasks Figure 1 suggests quite a different story.

2 Tiny Language Models

Weights of a 3B model in BF16/FP16 consume 6–7GB of memory (see Table 1 or Appendix A). The dominant memory tiers of common consumer hardware such as iPhone 16/17 or an RTX 5060, sit at 8GB (see Table 2 or Appendix A). Only LLMs with ≤3B parameters leave meaningful headroom for an operating system, KV cache, and activations at usable context lengths whereas a 4B model (∼8GB weights) would effectively exhaust common consumer hardware. Therefore:

> We define TLMs as models that can be effectively deployed on mainstream consumer devices, typically requiring less than 8 GB of memory at BF16/FP16 precision. This requirement is met by LMs with size below or around 3 billion parameters.

Prior work defines Small Language Models (SLMs) with somewhat looser and unclear upperbound (Lu et al., 2024a). We argue that there is a need for defining the tighter ∼3B parameters class (i.e., TLMs). This size is the maximum that effectively fits in standard BF16/FP16 precision on common consumer hardware that leaves room for normal device operation. This threshold aligns with the leading on-device deployments such as Apple Intelligence's foundation model (Gunter et al., 2024; Apple, 2025) or Meta's Llama 3.2 lightweight tier (Grattafiori et al., 2024; Meta AI, 2024).

Model	FP32	BF16 / FP16	FP8
Qwen3-0.6B (Qwen Team, 2025)	3.0GB	1.5GB	0.8GB
Llama-3.2-1B (Meta AI, 2024)	5.0GB	2.5GB	1.2GB
Qwen3-1.7B (Qwen Team, 2025)	6.9GB	3.4GB	1.7GB
Llama-3.2-3B (Meta AI, 2024)	12.8GB	6.4GB	3.2GB
Qwen3-4B (Qwen Team, 2025)	16.1GB	8.1GB	4.0GB
Llama-3.1-8B (Grattafiori et al., 2024)	32.1GB	16.1GB	8.0GB
Qwen3-8B (Qwen Team, 2025)	32.8GB	16.4GB	8.2GB

Table 1: GPU memory requirements for model weights at each precision. Values are based on reported Hugging Face safetensor shard sizes.

Table 3 shows that utilization of TLMs in zero-shot settings results in poor performance on our benchmarks (often close to random). Second, the table showcases that different viable approaches to evaluation (e.g., probability of the next token versus log-probability of the whole sequence) lead to dramatic gaps in performance. Third, fine-tuning TLMs for the specific tasks leads to superior performance (∼+25%) that can be on par with zero-/few-shot versions of frontier LLMs. These observations motivated us to systematically investigate several approaches to fine-tuning and evaluation in TLMs. The study provides rich basis for rethinking of evaluation and progress in this space.

3 Related Work

Defining TLMs in the SLM landscape. The boundary between *Small Language Models* (SLMs) and larger LLMs is not standardized. Surveys note that SLMs are usually motivated by lower latency, reduced cost, customization, privacy, and deployment in resource-constrained settings (Wang et al.,

Memory	Smartphones	Laptops / GPUs
8GB	iPhone 16 / 16 Plus / 16 Pro / 16 Pro Max; iPhone 17	RTX 4060 / 4070; RTX 5050 / 5060 / 5070 (8GB)
12GB	iPhone 17 Pro / Pro Max / Air; Galaxy S26 Ultra (256GB / 512GB)	RTX 4080; RTX 5070 (12GB) / 5070 Ti
16GB	Pixel 10 Pro / Pro XL; Galaxy S26 Ultra (1TB)	RTX 4090; RTX 5080; MacBook Air baseline (unified)
16–32GB	–	MacBook Air configurations, 16–32GB unified
24–32GB	–	RTX 5090 (24GB laptop / 32GB desktop)
>32GB	–	High-end MacBook Pro M-Max / M-Ultra, 64–128GB unified **(premium price tier only)**

Table 2: Representative memory budgets of commodity consumer devices (2025–2026).

Dataset	Model	Random	Zero-shot		CLS SFT	Δ
			LABEL-GEN	LOG-PROB		
HellaSwag	Qwen3-0.6B	25.0	25.58	36.27	**80.95**	+44.68
	Qwen3-1.7B	25.0	59.42	44.58	**90.81**	+31.39
	Qwen3-8B	25.0	80.11	55.21	**95.62**	+15.51
WinoGrande	Qwen3-0.6B	50.0	49.58	55.97	**70.80**	+14.83
	Qwen3-1.7B	50.0	51.39	57.10	**78.16**	+21.06
	Qwen3-8B	50.0	61.01	65.42	**89.53**	+24.11
PIQA	Qwen3-0.6B	50.0	51.47	67.41	**77.80**	+10.39
	Qwen3-1.7B	50.0	71.60	72.47	**85.04**	+12.57
	Qwen3-8B	50.0	86.51	76.28	**91.19**	+4.68
SciQ	Qwen3-0.6B	25.0	49.20	67.60	**90.70**	+23.10
	Qwen3-1.7B	25.0	88.30	81.90	**94.20**	+5.90
	Qwen3-8B	25.0	95.20	90.30	**96.00**	+0.80
ARC-Challenge	Qwen3-0.6B	25.0	27.55	31.67	**63.61**	+31.94
	Qwen3-1.7B	25.0	73.13	39.91	**79.57**	+6.44
	Qwen3-8B	25.0	90.04	55.11	**91.07**	+1.03
Mean (0.6B)		35.0	40.68	51.78	76.77	**+25.0**
Mean (1.7B)		35.0	68.77	59.19	85.56	**+15.5**
Mean (8B)		35.0	82.55	68.46	92.68	**+9.2**

Table 3: Zero-shot accuracy of Qwen3 models under two inference protocols compared with our LoRA classification-head CLS SFT. LABEL-GEN uses next-token argmax over answer-letter tokens. LOGPROB selects the candidate with the highest sequence log-probability.

2024). However, the parameter ranges differ substantially across papers: Srivastava et al. (2025) consider SLMs up to roughly 30B parameters, Lu et al. (2024b) focus on decoder-only models from 100M to 5B parameters, and Gupta et al. (2025) study SLMs mainly in the 1B–9B range. These definitions are all motivated by practical deployment on consumer devices such as phones, laptops, and edge hardware, where local execution can reduce cloud dependence and improve privacy (Yuan et al., 2023; Dubiel et al., 2024). In this work, we use the memory constraints of such devices as the main motivation for drawing a tighter boundary.

The term *TinyLLM / TinyLM / TLM* is even less consistent. Recent edge-agent work studies sub-3B SLMs for agentic tasks while using TinyLLM and SLM terminology closely together (Haque et al., 2025). Other work uses the TinyLLM name for models as small as 30M–120M parameters (Kandala et al., 2024), while "tiny language model" work also appears at the 1B–1.5B scale (Tang et al., 2024). Related surveys of tiny language models for automation and control do not provide a single parameter cutoff for TinyLLMs (Lamaakal et al., 2025). We therefore define **TLMs** operationally as models below 3B parameters. This cutoff separates them from the broader SLM category, matches common open-weight size tiers, and reflects the practical goal that the model weights should fit on many consumer devices with enough remaining memory for normal execution. Appendix A lists representative model families and device memory capacities supporting this choice in detail.

Prediction-interface sensitivity in multi-choice evaluation. Multi-choice evaluation of LLMs is highly sensitive to how the answer is extracted from the model. Lyu et al. (2024) show that probability-based scoring and free-form generation often disagree on the selected option, and Molfese et al. (2025) find that common extraction protocols misalign with human judgement, especially when models produce rationales before committing to a letter. Even tokenisation details matter; Sanz-Guerrero et al. (2025) report up to 11% accuracy swings, and reshuffled model rankings, depending solely on whether the space after "Answer:" is tokenised with or apart from the answer letter. More broadly, Polo et al. (2024) argue that single-prompt scores are unstable and propose estimating a distribution over prompts. These findings push us to rethink how progress is measured on multi-choice benchmarks, and motivate our treatment of the *training-and-evaluation interface* as a first-class object. Rather than treating instruction-formatted label generation, LM-head log-probability scoring, and classification-head candidate scoring as interchangeable, we evaluate them as distinct prediction strategies. As our results show, the gap between them is substantial for TLMs, where switching from the trained interface to a different scoring interface can materially change reported accuracy.

Fine-tuning approaches for multi-choice tasks. The strongest published fine-tuned reference on the commonsense benchmarks we consider is UNICORN (Lourie et al., 2021), which adapts T5-11B (Raffel et al., 2020) via multitask transfer on RAINBOW (αNLI, CosmosQA, HellaSwag, PIQA, SocialIQa, WinoGrande) followed by per-task fine-tuning, scoring candidates by

generation likelihood at test time, structurally similar to the log-probability scoring used by `lm-evaluation-harness` (Gao et al., 2024) and to our LP-Sum metric. Our work differs along three axes. We use decoder-only Qwen3 models 1.4 to 19× smaller than T5-11B, train a single-task LoRA adapter per benchmark with no multitask transfer, and replace generative candidate likelihood with a discriminative classification head.Yousefiramandi and Cooney (2025) also study classification-head fine-tuning for decoder-only LLMs, comparing an embedding-based classifier against instruction-formatted label generation under LoRA/QLoRA, and show that classifier can outperform label generation on text classification. Our work is complementary but studies a different setting. Rather than assigning a label to a single input document, we reformulate verifiable multiple-choice tasks as candidate-sequence ranking

4 Method

We study Classification-head SFT, a discriminative alternative to instruction-formatted label-generation supervised fine-tuning for multiple-choice benchmarks. Instead of fine-tuning the model to generate an answer option in response to a prompt, we fine-tune it to assign a scalar score to each candidate input. The prediction is then obtained by selecting the highest-scoring candidate.

4.1 Constructing Candidate Sequences

Our formulation applies to any task where each data point has a single correct output and a finite set of incorrect alternatives. The central requirement is that each candidate and its context can be converted into an input sequence that can be scored by the model. Thus, instead of fine-tuning the model to generate the answer, we transform the task into a candidate-scoring problem.

Formally, for an input instance x with candidate outputs $\{y_1, \ldots, y_K\}$ and gold index y_i, we construct K candidate sequences:

$$s_k = f(x, y_k),$$

where f is a task-specific input transformation function. The same scoring model is then applied to each s_k, producing one scalar score z_k per candidate. This gives a unified fine-tuning format across tasks whose surface forms may differ (e.g., sentence completion, fill-in-the-blank, or question answering). This candidate-sequence framing connects directly to log-probability-based multiple-choice inference (Brown et al., 2020b; Lyu et al., 2024), in which each candidate is scored independently. Our method differs in what does the scoring: a single linear scoring head over the final hidden state, not a per-token language-modeling head over the candidate's tokens.

The form of f depends on the task structure: it concatenates context with candidate for completion tasks (HellaSwag), substitutes a candidate into a blank for fill-in-the-blank tasks (WinoGrande), and pairs a question with each candidate answer for QA tasks (PIQA, ARC-Challenge, SciQ). Table 4 shows f on a HellaSwag example; the per-dataset templates are given in Appendix B.

4.2 Classification head architecture

We implement the candidate scoring model by loading the pretrained decoder-only transformer backbone and replacing its language-modeling head with a sequence-level scoring head.[1] Whereas the standard LM head is a linear projection $\mathbf{W}_{\text{LM}} \in \mathbb{R}^{d \times |V|}$ that produces a distribution over the full vocabulary at every position, our scoring head is a bias-free linear projection $w \in \mathbb{R}^{d \times 1}$ applied only to the final non-padding hidden state of the candidate sequence. The head contains no intermediate hidden layer and applies no non-linear activation.

Formally, for a minibatch of B examples with K candidates per example, we tokenize and pad all candidate sequences to length L, obtaining tensors of shape $[B, K, L]$. We then flatten the candidate dimension and run the model on the resulting batch of shape $[BK, L]$. Let $h_{i,k} \in \mathbb{R}^d$ denote the final non-padding hidden state for candidate k of example i. The classification head computes

$$z_{i,k} = w^\top h_{i,k},$$

where $w \in \mathbb{R}^d$ is the learned scoring vector. For example, Qwen3-0.6B has an embedding size of $d = 1024$, so the classification head for it adds only 1024 trainable parameters when `num_labels=1`.

4.3 Fine-tuning Objective

Given a gold answer index $y_i \in \{1, \ldots, K\}$ for example i, we treat the K candidate scores as logits over the answer choices. For a minibatch of B examples, the model produces a score matrix $Z \in$

[1] We use the `AutoModelForSequenceClassification` interface from Hugging Face (Wolf et al., 2020) with `num_labels=1`.

CLS: candidate-sequence scoring	Label-Gen SFT: answer-label generation	Gold-Only SFT: gold-only continuation
All candidates are scored independently: A. Roof shingle removal: A man is sitting on a roof. He is using wrap to wrap a pair of skis. B. Roof shingle removal: A man is sitting on a roof. He is ripping level tiles off. C. Roof shingle removal: A man is sitting on a roof. He is holding a Rubik's cube. D. Roof shingle removal: A man is sitting on a roof. He **starts pulling up roofing on a roof.** *Training: cross-entropy over four scalar candidate scores.*	`<|im_start|>user` Choose the most plausible ending. Context: Roof shingle removal: A man is sitting on a roof. He A. is using wrap to wrap a pair of skis. B. is ripping level tiles off. C. is holding a Rubik's cube. D. starts pulling up roofing on a roof. Answer with only one letter: A, B, C, or D. Answer:`<|im_end|>` `<|im_start|>assistant` **D** *Training: next-token prediction on the answer label only.*	*Only the gold continuation is used:* Roof shingle removal: A man is sitting on a roof. He **starts pulling up roofing on a roof.** *Training: next-token prediction on the gold ending tokens only; context tokens are masked from the loss. At evaluation, all candidate endings are scored by LM log-probability.*

Table 4: Example training formats for one HellaSwag instance. Blue text denotes the shared context, orange denotes incorrect candidate endings, green denotes the gold ending, and purple denotes the supervised answer label. The three columns correspond to our three training paradigms. CLS directly compares all candidate sequences, Label-Gen SFT trains the model to emit an answer label, and Gold-Only SFT trains only on the gold continuation.

$\mathbb{R}^{B\times K}$, where $z_{i,k}$ is the scalar score for candidate k of example i. We minimize the average cross-entropy loss over the minibatch:

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(z_{i,y_i})}{\sum_{k=1}^{K}\exp(z_{i,k})}.$$

This objective trains the model discriminatively over the candidates for the same example. The softmax is therefore applied only across the K candidate scores during the loss computation, not inside the model itself. In logit space, the gradient raises the score assigned to the gold candidate and suppresses the scores of incorrect candidates. This differs from Label-Gen SFT, where the model is fine-tuned to generate an answer label or answer tokens given a prompt. In our formulation, the candidate texts themselves are directly compared through their learned sequence-level scores.

5 Experiments

5.1 Datasets

Our experiments use five English multi-choice benchmarks grouped into two tiers. The three **primary commonsense reasoning benchmarks**: HellaSwag, WinoGrande, and PIQA are widely reported in language-model evaluations (Brown et al., 2020b; Grattafiori et al., 2024) and remain non-trivial even for strong models. Together they cover a wide range of complementary forms of commonsense reasoning (event continuation, coreference resolution, physical commonsense). These benchmarks also reflect suitable use cases for on-device TLMs such as choosing the next likely action in an assistant workflow or resolving what a short message refers to. The additional two **science reasoning benchmarks**: SciQ and ARC-Challenge are memorisation-intensive question-answering tasks that depend more on stored factual knowledge than on candidate discrimination. We include them to test whether the candidate-scoring advantage transfers beyond commonsense reasoning.

Dataset	Type	Choices	Train	Eval
Primary commonsense reasoning benchmarks				
HellaSwag	Completion	4	39,905	10,042
WinoGrande (xl)	Fill-in-the-blank	2	40,398	1,767
PIQA	Physical QA	2	16,113	1,838
Memorization intensive science benchmarks				
SciQ	Science QA	4	11,679	1,000
ARC-Chal	Science QA	4	1,119	1,172

Table 5: Dataset statistics. Note that ARC-Challenge has a significantly smaller training set (1.1k) than the other four benchmarks.

HellaSwag (Zellers et al., 2019) is a 4-way sentence-completion task with adversarially filtered distractors that remain locally fluent and grammatically compatible with the context, so that simple surface cues are insufficient and commonsense event understanding is required. The model must decide which continuation is coherent with the situation described by the context.

WinoGrande (Sakaguchi et al., 2020) is a 2-way fill-in-the-blank coreference task at scale, adversarially filtered to suppress annotation artifacts and word-association shortcuts from pretraining. We use the `winogrande_xl` version. Correct resolution typically depends on implicit commonsense understanding rather than syntax alone.

PIQA (Bisk et al., 2020) is a 2-way physical-commonsense dataset in which two lexically similar candidate solutions are paired with an everyday goal. Solving it requires reasoning about object affordances, materials, and practical physical constraints in the physical world.

SciQ (Welbl et al., 2017) is a 4-way crowd-sourced science multiple-choice benchmark spanning physics, chemistry, biology, and earth science. We use it without the support passages.

ARC-Challenge (Clark et al., 2018) is a 4-way grade-school science question-answering benchmark, restricted to the "Challenge" set of items that neither a retrieval baseline nor a word co-occurrence baseline can solve.

5.2 Models

We evaluate four open-weight decoder-only models from the Qwen3 family (Qwen Team, 2025) spanning **0.6B**, **1.7B**, **4B**, and **8B** parameters. The first two fall within our TLM regime ($\leq$3B parameters), while the 4B and 8B models let us test whether the same trends persist beyond it. This set provides a controlled scaling axis within a recent, competitive open-weight family.

All models are loaded and fine-tuned in `bfloat16`. We disable Qwen3's thinking mode throughout all experiments so that models are fine-tuned and evaluated to produce the final answer, without generating intermediate reasoning. For Label-Gen SFT, we use the default Hugging Face chat template associated with each model.

5.3 Experimental Design

For HellaSwag and PIQA, we use the public validation split as the evaluation set because test labels are private, otherwise we use the available test split. All the datasets share a common verifiable structure: each example contains a finite set of candidate answers and exactly one correct choice. With the candidate-sequence formulation introduced in Section 4.1, they provide a meaningful testbed for studying fine-tuning strategies for TLMs.

For each dataset and model, we compare three fine-tuning paradigms under a unified training configuration so that the only difference between conditions is the fine-tuning objective.

Fine-tuning Objectives. We evaluate the following three paradigms.

- **CLS: Classification-head SFT.** This is our proposed method. Each candidate sequence is scored by a sequence-level classification head, and the model is trained with cross-entropy over the candidate scores.
- **Label-Gen SFT: Instruction-formatted label-generation supervised fine-tuning.** The model receives a chat-style multiple-choice prompt containing all answer options and is trained to generate the correct answer letter, such as " A" or " B" (Sanz-Guerrero et al., 2025). The loss is applied only to the target answer token.
- **Gold-Only SFT: Gold-only continuation supervised fine-tuning.** The model is trained only on the gold context-candidate sequence in the same format as CLS-SFT using the standard next-token language-modeling objective, with context tokens masked from the loss. This tests whether increasing the likelihood of the correct continuation is sufficient, without explicitly training on incorrect candidates.

Training setup. All three paradigms train LoRA adapters (Hu et al., 2022) on a frozen Qwen3 backbone for 2 epochs at learning rate 2×10^{-4}, with per-device batch size 8, gradient accumulation over 8 steps. Each fine-tuning run (with evals) took roughly 12–16 hours on a single NVIDIA L4 GPU. Full LoRA and optimizer hyperparameters are in Appendix C.1.

Evaluation protocols. We evaluate each training paradigm in its native prediction interface: CLS uses classification-head argmax over candidate scores, Label-Gen SFT uses next-token answer-label accuracy, and Gold-Only SFT uses LM log-probability scoring over candidate continuations. We also evaluate the Label-Gen SFT adapter as a candidate scorer by applying LM log-probability scoring to the same candidate sequences used by CLS. We denote this diagnostic evaluation as Label-Gen LogProb. Additional diagnostics including the LP-Full/LP-Sum/LP-Avg/LP-Char metrics, are described in Appendix C.2.

Dataset	TLM	Qwen3 Params	CLS-SFT	Label-Gen SFT	Δ	Label-Gen LogProb	Gold-Only SFT	p
HellaSwag	✓	0.6B	**80.95**	78.50	+2.45	47.93	52.98	$< .001^{**}$
	✓	1.7B	**90.81**	89.33	+1.48	59.59	66.75	$< .001^{**}$
		4B	94.24	93.90	+0.34	68.29	75.48	.105
		8B	95.62	95.51	+0.11	74.81	80.15	.598
WinoGrande	✓	0.6B	**70.80**	67.80	+3.00	56.82	57.61	$.006^{**}$
	✓	1.7B	78.16	77.48	+0.68	62.37	65.53	.518
		4B	87.21	87.15	+0.06	66.67	67.74	1.000
		8B	89.53	89.25	+0.28	69.89	72.21	.733
PIQA	✓	0.6B	**77.80**	74.70	+3.10	68.34	71.87	$.003^{**}$
	✓	1.7B	**85.04**	82.59	+2.45	74.86	77.48	$.006^{**}$
		4B	89.66	89.39	+0.27	76.93	79.98	.754
		8B	91.19	92.11	−0.92	79.92	82.05	.201
SciQ	✓	0.6B	**90.70**	88.50	+2.20	75.60	86.00	$.026^{*}$
	✓	1.7B	94.20	93.20	+1.00	82.80	92.40	.203
		4B	95.00	95.10	−0.10	90.80	93.00	1.000
		8B	96.00	96.90	−0.90	92.20	95.80	.188
ARC-Challenge	✓	0.6B	63.61	64.03	−0.42	36.39	42.75	.817
	✓	1.7B	79.57	79.66	−0.09	43.09	55.71	1.000
		4B	88.33	89.70	−1.37	56.31	64.03	.178
		8B	91.07	92.45	−1.38	58.88	68.84	.121

Table 6: Test accuracy on all five benchmarks. The upper block is the three primary commonsense benchmarks; the lower block is the two memorisation-intensive science benchmarks. CLS-SFT and LABEL-GEN SFT report native trained-format accuracy; Label-Gen LogProb and Gold-Only SFT report normalised log-probability accuracy on the candidate-sequence format (LP-Char for HellaSwag, PIQA, ARC-Challenge and SciQ, and LP-Avg for WinoGrande). McNemar p-values compare paired correctness of CLS-SFT and Label-Gen SFT; significant entries are bolded $^{*}p < 0.05$, $^{**}p < 0.01$. Underlined entries mark the higher of CLS-SFT vs. Label-Gen SFT per row.

6 Results and Discussion

Table 6 reports test accuracies on all five benchmarks across the four Qwen3 model sizes and three fine-tuning paradigms. Figure 1 situates the commonsense results against prior external baselines. Detailed log-prob evaluations (LP-Full, LP-Sum, LP-Avg, LP-Char) are reported in Appendix C.3.

6.1 Classification SFT is superior at tiny scale

The clearest pattern is that **Classification-head SFT is most beneficial for TLMs**. At 0.6B, CLS-SFT improves over Label-Gen SFT by +2.45, +3.00, +3.10, and +2.20 points on HellaSwag, WinoGrande, PIQA, and SciQ; at 1.7B the gains remain positive at +1.48, +0.68, +2.45, and +1.00. ARC-Challenge is the only TLM-scale exception, and the difference is negligible (−0.42 at 0.6B, −0.09 at 1.7B, both non-significant with $p \geq 0.8$). We attribute this to its small training split of only 1,119 examples. We interpret the broader pattern as evidence that **tiny models benefit from being trained to compare candidate sequences directly**. Under Label-Gen SFT the supervision is concentrated on a single answer-letter token, whereas CLS-SFT assigns an explicit scalar score to each candidate sequence and optimises a softmax over those scores, directly exposing the model to both gold and distractor candidates during training.

6.2 Commonsense reasoning benchmarks

On HellaSwag, WinoGrande, and PIQA, CLS-SFT is the higher of the two finetuning methods in every (dataset, model) row across all model scales, except PIQA-8B where Label-Gen SFT is better but not significant (91.19 vs. 92.11, $\Delta = -0.92$, $p = .201$). At the TLM scale the advantage is also statistically robust: five of the six TLM McNemar comparisons reach $p < 0.01$ under the exact binomial test, with WinoGrande-1.7B ($p = .518$) the only exception. **To the best of our knowledge, the CLS-SFT accuracies reported here for Qwen3-0.6B and 1.7B are the highest published for these models on HellaSwag, WinoGrande, and PIQA.**

6.3 Memorization intensive benchmarks

The science benchmarks split into two complementary cases. On SciQ, which has a large training pool

(11.7k examples) and a clear knowledge-retrieval flavor, CLS-SFT significantly outperforms Label-Gen SFT at 0.6B (90.70% vs. 88.50%, $p = .026$) and remains the higher-scoring method at 1.7B, mirroring the TLM-scale pattern from the commonsense block. On ARC-Challenge, Label-Gen SFT is directionally higher at every scale, although none of the differences is significant ($p \geq 0.12$ everywhere). We attribute this to two compounding factors. First, ARC-Challenge has only 1,119 training examples, which limits how much either fine-tuning objective can reshape behaviour. Second, the benchmark is gated by retrieval of stored scientific facts rather than by discrimination among plausible candidate texts. If the model does not possess the relevant fact or concept, changing the fine-tuning objective may not help as much.

6.4 Classification SFT at larger scale

The advantage of CLS-SFT shrinks sharply once we move past the TLM regime. On the commonsense block, CLS-SFT is still directionally ahead at 4B and 8B mostly, but none of the 4B or 8B differences is statistically significant. The single directional loss on commonsense is PIQA-8B. On the science block, the pattern flips at scale: Label-Gen SFT is directionally higher than CLS-SFT at 4B and 8B on both SciQ and ARC-Challenge, but again every one of those differences is non-significant. The narrowing therefore reflects **larger models becoming better able to use the standard instruction-style multiple-choice format**. Once the model has enough capacity to reliably map the candidate texts in the prompt to the answer-label token, the additional structure of an explicit candidate-scoring objective stops mattering.

6.5 Gold-Only continuation SFT

Across all five benchmarks the Gold-Only baseline trails both CLS-SFT and Label-Gen SFT, often by ten points or more (e.g., HellaSwag-1.7B: 66.75% vs. 90.81% for CLS-SFT). Although Gold-Only SFT is the closest of the three paradigms to the model's original pretraining objective (next-token prediction on the gold context-candidate sequence, with the context masked from the loss), it never directly supervises against distractors, leaving the model with no signal for ranking wrong candidates below the right one. For verifiable multi-choice tasks, **supervising the contrast between gold and distractors is more valuable than additional density on the gold candidate.**

6.6 Fine-tuning objective determines the prediction interface

Our results also show that **fine-tuning changes which prediction interface works best**. The same backbone can be used in three ways (answer-letter generation through the LM head, candidate log-probability scoring through the LM head, or sequence-level scoring through a classification head), and after fine-tuning these interfaces are no longer interchangeable.

When a CLS-SFT-trained model is evaluated through the LM head instead of its native classification head, accuracy drops sharply (e.g., Qwen3-1.7B+CLS-SFT on HellaSwag falls from 90.81% to 59.15% under LP-Char; on PIQA, from 85.04% to 70.24%). The complementary swing appears for Label-Gen SFT: re-evaluating the same adapter as a candidate scorer in the CLS-SFT-style input format (Label-Gen LogProb) costs roughly 25 points on HellaSwag-4B (93.90% → 68.29%) and about 20 points on WinoGrande-8B (89.25% → 69.89%).

The fine-tuning objective and the evaluation interface are therefore tightly coupled. Evaluating a fine-tuned model through a different interface can substantially underestimate the capability learned by that objective, and the swings are largest for tiny models, where the choice of interface can shift reported accuracy by tens of points.

7 Conclusion

We define Tiny Language Models, TLMs - as sub-3B language models motivated by practical consumer-device memory constraints, and study how this regime should be fine-tuned for verifiable multiple-choice reasoning. On a unified LoRA setup across Qwen3 0.6B–8B, our classification-head formulation (CLS-SFT) beats instruction-formatted label generation by +2 to +3 points on every commonsense benchmark at 0.6B and 1.7B (five of six small-scale comparisons significant at $p < 0.01$), extends to SciQ at the TLM scale, and is statistically neutral on ARC-Challenge. To our knowledge, the resulting Qwen3-0.6B and Qwen3-1.7B numbers on HellaSwag, WinoGrande, and PIQA are the highest published for these models.

For practitioners deploying TLMs on consumer hardware, our results offer a concrete recipe: classification-head fine-tuning on a modest LoRA budget leads to best performance. This makes the approach a safe default for verifiable multi-choice tasks with models at TLM scale.

Limitations

While we aim to provide a rigorous and controlled comparison of fine-tuning interfaces for TLMs, our study necessarily makes several scope and design choices. We acknowledge the following limitations.

- **Scoring-head capacity.** We use the standard `AutoModelForSequenceClassification` head, which is a single linear projection from the final non-padding hidden state to one scalar score. We do not add an additional MLP or non-linear activation. This is appropriate for cross-entropy training and is the simplest approach, which operates on raw logits, but a deeper scoring head may provide additional capacity and could improve accuracy. We leave this for future work.
- **Prompt-format sensitivity.** We do not exhaustively explore prompt and answer-format choices. For the Label-Gen SFT baseline, we use answer letters such as A, B, C, and D. Other formats, including numbered options, different chat templates, or dataset-specific instructions, may affect the strength of the SFT baseline.
- **No few-shot inference.** Our experiments focus on fine-tuned evaluation without few-shot examples at inference time. Few-shot prompting may change the relative performance of Label-Gen SFT and candidate-sequence scoring.
- **No reasoning supervision.** Recent small language models increasingly support reasoning-style generation, but our experiments do not use chain-of-thought supervision or reasoning traces. Our goal is to isolate the effect of the candidate-scoring objective, but combining it with reasoning-oriented training or inference may yield stronger results.
- **Limited model-family coverage.** We focus on the Qwen3 model family to study scaling trends across a consistent set of checkpoints. Due to compute constraints, we do not repeat the full comparison across multiple model families.
- **Limited scale beyond 8B.** We evaluate models up to the 7–8B parameter range. It remains unclear whether the classification-head advantage persists for much larger models, where Label-Gen SFT may already be strong enough to exploit the multiple-choice prompt format.
- **Limited task and language coverage.** Our evaluation focuses on English multiple-choice benchmarks, with additional science-domain results reported separately. We do not evaluate multilingual tasks, open-ended generation tasks, or verifiable tasks where candidate sets must be generated rather than provided by the dataset.

Ethical Considerations

This study evaluates publicly available open-weight language models on standardised, publicly available multi-choice benchmarks. No private, sensitive, or human-subjects data were collected or used at any stage of the work. All training and evaluation details, including hyperparameters, prompt templates, and per-dataset configurations, are reported in the main paper and appendices to support full reproducibility.

AI Assistance: We used Claude Code and ChatGPT for drafting LaTeX table code, refining wording, and suggesting code-organization patterns for the experimental pipeline. All generated content was reviewed and revised thoroughly by the authors to ensure accuracy and clarity.

References

Apple. 2025. Apple Intelligence foundation language models: Tech report 2025. *arXiv preprint arXiv:2507.13575*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020b. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mateusz Dubiel, Yasmine Barghouti, Kristina Kudryavtseva, and Luis A. Leiva. 2024. On-device query intent prediction with lightweight llms to support ubiquitous conversations. *Scientific Reports*, 14(12731).

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation. *Preprint*, arXiv:2405.14782.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, and 1 others. 2024. Apple Intelligence foundation language models. *arXiv preprint arXiv:2407.21075*.

Akanksha Gupta, Bijo Thomas, Harshita Asnani, Phanindra Reddy Madduru, Samia Feroze, Shreyas Subramanian, Vikram Elango, and Mecit Gungor. 2025. Small language models (slms) can still pack a punch: A survey. *arXiv preprint arXiv:2501.05465*.

Mohd Ariful Haque, Fahad Rahman, Kishor Datta Gupta, Khalil Shujaee, and Roy George. 2025. TinyLLM: Evaluation and optimization of small language models for agentic tasks on edge devices. *arXiv preprint arXiv:2511.22138*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Savitha Viswanadh Kandala, Pramuka Medaranga, and Ambuj Varshney. 2024. TinyLLM: A framework for training and deploying language models at the edge computers. *arXiv preprint arXiv:2412.15304*.

Ismail Lamaakal, Yassine Maleh, Khalid El Makkaoui, Ibrahim Ouahbi, Paweł Pławiak, Osama Alfarraj, May Almousa, and Ahmed A. Abd El-Latif. 2025. Tiny language models for automation and control: Overview, potential applications, and future research directions. *Sensors*, 25(5):1318.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25366–25394.

Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. UNICORN on rainbow: A universal commonsense reasoning model on a new multitask benchmark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13480–13488.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. 2024a. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. 2024b. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*.

Chenyang Lyu, Minghao Wu, and Alham Fikri Aji. 2024. Beyond probabilities: Unveiling the misalignment in evaluating large language models. *Preprint*, arXiv:2402.13887.

Meta AI. 2024. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`

Francesco Maria Molfese, Luca Moroni, Luca Gioffrè, Alessandro Scirè, Simone Conia, and Roberto Navigli. 2025. Right answer, wrong score: Uncovering the inconsistencies of LLM evaluation in multiple-choice question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18477–18494. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. 2024. Efficient multi-prompt evaluation of LLMs. *Preprint*, arXiv:2405.17202.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Mario Sanz-Guerrero, Minh Duc Bui, and Katharina von der Wense. 2025. Mind the gap: A closer look at tokenization for multiple-choice question answering with LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. 2025. ThinkSLM: Towards reasoning in small language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32612–32662. Association for Computational Linguistics.

Yehui Tang, Fangcheng Liu, Yunsheng Ni, Yuchuan Tian, Zheyuan Bai, Yi-Qi Hu, Sichao Liu, Shangling Jui, Kai Han, and Yunhe Wang. 2024. Rethinking optimization and architecture for tiny language models. In *Proceedings of the 41st International Conference on Machine Learning*.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, TzuHao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. 2024. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *arXiv preprint arXiv:2411.03350*.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Amirhossein Yousefiramandi and Ciarán Cooney. 2025. Fine-tuning causal llms for text classification: Embedding-based vs. instruction-based approaches. *Preprint*, arXiv:2512.12677.

Jinliang Yuan, Chen Yang, Dongqi Cai, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, and 1 others. 2023. Mobile foundation model as firmware. *arXiv preprint arXiv:2308.14363*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800. Association for Computational Linguistics.

A Model and Device Memory

This appendix provides the hardware and memory context motivating our focus on small language models. Table 7 reports approximate model-weight memory requirements under common numerical formats, while Table 8 summarizes representative consumer-device memory capacities. These estimates are intended only as order-of-magnitude constraints: they exclude activations, optimizer states, framework overhead, and KV cache.

Family	Model tier	Params	FP32	BF16/FP16	FP8
Qwen family					
Qwen2.5	0.5B	0.5B	2.0GB	1.0GB	0.5GB
Qwen3	0.6B	0.6B	3.0GB	1.5GB	0.8GB
Qwen2.5	1.5B	1.5B	6.0GB	3.0GB	1.5GB
Qwen3	1.7B	1.7B	6.9GB	3.4GB	1.7GB
Qwen2.5	3B	3.0B	12.0GB	6.0GB	3.0GB
Qwen3	4B	4.0B	16.1GB	8.1GB	4.0GB
Qwen2.5	7B	7.0B	28.0GB	14.0GB	7.0GB
Qwen3	8B	8.0B	32.8GB	16.4GB	8.2GB
Llama family					
Llama-3.2	1B	1.0B	5.0GB	2.5GB	1.2GB
Llama-3.2	3B	3.0B	12.8GB	6.4GB	3.2GB
Llama-3/3.1	8B	8.0B	32.1GB	16.1GB	8.0GB
Gemma family					
Gemma-3	1B	1.0B	4.0GB	2.0GB	1.0GB
Gemma	2B	2.0B	8.0GB	4.0GB	2.0GB
Gemma-2	2B	2.0B	8.0GB	4.0GB	2.0GB
Gemma-3	4B	4.0B	16.0GB	8.0GB	4.0GB
Gemma-2	9B	9.0B	36.0GB	18.0GB	9.0GB
Phi family					
Phi	1.5	1.3B	5.2GB	2.6GB	1.3GB
Phi	2	2.7B	10.8GB	5.4GB	2.7GB
Phi	3.5-mini	3.8B	15.2GB	7.6GB	3.8GB
Phi	4-mini	3.8B	15.2GB	7.6GB	3.8GB
Phi	3-small	7.0B	28.0GB	14.0GB	7.0GB
Other open small-model families					
MobileLLM	0.6B	0.6B	2.4GB	1.2GB	0.6GB
MobileLLM	1.5B	1.5B	6.0GB	3.0GB	1.5GB
OpenELM	1.1B	1.1B	4.4GB	2.2GB	1.1GB
OpenELM	3B	3.0B	12.0GB	6.0GB	3.0GB
SmolLM2	1.7B	1.7B	6.8GB	3.4GB	1.7GB
Mistral	7B	7.0B	28.0GB	14.0GB	7.0GB

Table 7: Approximate model-weight memory across common precision formats. For the Qwen3 and Llama rows used in our main experiments, values match the Hugging Face safetensor shard sizes reported in Table 1. Other rows are included as approximate family-level references. All values exclude activations, optimizer states, runtime overhead, and KV cache, so actual inference or training memory is higher.

Tier	Device / configuration	Memory	Type
8GB and below: common entry or mid-range COTS memory limits			
Phone	iPhone 17 base	8GB	Unified system memory
Phone	Google Pixel 8a	8GB	Unified system memory
Phone	Samsung Galaxy A56, lower configuration	8GB	Unified system memory
Phone	Redmi Note 14 Pro, lower configuration	8GB	Unified system memory
Phone	Nothing Phone (2a), lower configuration	8GB	Unified system memory
Laptop	HP Victus 15 with RTX 3050-class GPU	4GB VRAM	Dedicated GPU memory
Laptop	NVIDIA RTX 4050 laptop GPU systems	6GB VRAM	Dedicated GPU memory
Laptop	ASUS ROG Zephyrus G14 with RTX 4060	8GB VRAM	Dedicated GPU memory
Laptop	HP OMEN / Victus with RTX 4070	8GB VRAM	Dedicated GPU memory
Laptop	Dell XPS 16 with RTX 5070	8GB VRAM	Dedicated GPU memory
Laptop	NVIDIA RTX 5060 / 5070 laptop GPU systems	8GB VRAM	Dedicated GPU memory
Above 8GB: higher-end phones, UMA laptops, and gaming/creator laptops			
Phone	iPhone 17 Air / Pro / Pro Max	12GB	Unified system memory
Phone	Google Pixel 9	12GB	Unified system memory
Phone	Google Pixel 9 Pro / Pro XL	16GB	Unified system memory
Phone	Samsung Galaxy S25 Ultra	12GB	Unified system memory
Phone	Samsung Galaxy A56, higher configuration	12GB	Unified system memory
Phone	OnePlus 13	12GB / 16GB	Unified system memory
Phone	OnePlus Nord 4	12GB / 16GB	Unified system memory
Phone	Xiaomi 15 Ultra	16GB	Unified system memory
Phone	OPPO Find X8 Pro	16GB	Unified system memory
Phone	Nothing Phone (3)	12GB / 16GB	Unified system memory
Phone	Nothing Phone (2a), higher configuration	12GB	Unified system memory
Laptop	MacBook Air, Apple Silicon	16GB / 24GB / 32GB	Unified memory
Laptop	MacBook Pro, Apple Silicon Pro/Max	24GB–128GB	Unified memory
Laptop	Microsoft Surface Laptop 13.8" / 15"	16GB / 32GB / 64GB	System memory
Laptop	Dell XPS 13	16GB / 32GB	System memory
Laptop	Dell XPS 16	16GB / 32GB / 64GB	System memory
Laptop	HP OmniBook Ultra / AI PC class	16GB / 32GB	System memory
Laptop	HP Spectre / premium convertible class	16GB / 32GB	System memory
Laptop	ASUS ROG Zephyrus / similar creator laptops	16GB / 32GB	System memory plus optional dGPU VRAM
Laptop	NVIDIA RTX 4080 laptop GPU systems	12GB VRAM	Dedicated GPU memory
Laptop	NVIDIA RTX 4090 laptop GPU systems	16GB VRAM	Dedicated GPU memory
Laptop	NVIDIA RTX 5070 Ti laptop GPU systems	12GB VRAM	Dedicated GPU memory
Laptop	NVIDIA RTX 5080 laptop GPU systems	16GB VRAM	Dedicated GPU memory
Laptop	NVIDIA RTX 5090 laptop GPU systems	24GB VRAM	Dedicated GPU memory

Table 8: Representative memory capacities of recent consumer devices. Phone and Apple-silicon laptop memory is unified and shared with the operating system, applications, model weights, activations, and KV cache. NVIDIA laptop GPUs use dedicated VRAM. **For laptops with discrete GPUs, system RAM is not a substitute for GPU VRAM for practical local LLM inference: the model must fit primarily in GPU memory to run at usable speed.** Apple-silicon unified memory is a partial exception because the CPU and GPU share the same memory pool, but it is still shared with the rest of the system. These constraints motivate the sub-3B TLM regime: larger 4B–8B models often require high-end memory configurations, aggressive quantization, or offloading.

B Input Formats and Training Templates

This appendix gives concrete examples and abstract templates for the three training paradigms. The examples illustrate how the same supervised item is converted into candidate-sequence scoring, answer-label generation, and gold-only continuation supervised fine-tuning. We include these details to make the comparison reproducible and to clarify exactly where the training loss is applied in each paradigm.

B.1 Concrete Input Examples

Table 9 shows one representative example from each primary commonsense benchmark. In CLS, every answer option is converted into a separate candidate sequence and scored with the classification head. In Label-Gen SFT, all choices are placed in a chat-template prompt and the model is supervised to generate the answer label. In Gold-Only SFT, only the gold candidate sequence is used for causal language-model fine-tuning.

Dataset	CLS: candidate-sequence scoring	Label-Gen SFT: answer-label generation	Gold-Only SFT: gold-only continuation
HellaSwag	*All candidates are scored independently:* A. Roof shingle removal: A man is sitting on a roof. He is using wrap to wrap a pair of skis. B. Roof shingle removal: A man is sitting on a roof. He is ripping level tiles off. C. Roof shingle removal: A man is sitting on a roof. He is holding a Rubik's cube. D. Roof shingle removal: A man is sitting on a roof. He **starts pulling up roofing on a roof.** *Training: cross-entropy over four scalar classification-head scores.*	`<|im_start|>user` Choose the most plausible ending. Context: Roof shingle removal: A man is sitting on a roof. He A. is using wrap to wrap a pair of skis. B. is ripping level tiles off. C. is holding a Rubik's cube. D. starts pulling up roofing on a roof. Answer with only one letter: A, B, C, or D. Answer:`<|im_end|>` `<|im_start|>assistant` **D** *Training: next-token prediction on the answer label.*	*Only the gold continuation is used:* Roof shingle removal: A man is sitting on a roof. He **starts pulling up roofing on a roof.** *Training: next-token prediction only on the gold ending tokens; context tokens are masked from the loss.*
WinoGrande	*Each option is substituted into the blank:* A. Sarah was a much better surgeon than Maria so Sarah always got the easier cases. B. Sarah was a much better surgeon than Maria so **Maria** always got the easier cases. *Training: cross-entropy over two scalar classification-head scores.*	`<|im_start|>user` Fill in the blank with the most appropriate option. Sentence: Sarah was a much better surgeon than Maria so _ always got the easier cases. A. Sarah B. Maria Answer with only one letter: A or B. Answer:`<|im_end|>` `<|im_start|>assistant` **B** *Training: next-token prediction on the answer label.*	*Only the gold substituted sentence is used:* Sarah was a much better surgeon than Maria so **Maria** always got the easier cases. *Training: next-token prediction on the substituted option and remaining suffix; prefix tokens are masked.*
PIQA	*Each solution is paired with the same goal:* A. Question: How do I ready a guinea pig cage for its new occupants? Answer: **Provide the guinea pig with a cage full of a few inches of bedding made of ripped paper strips, you will also need to supply it with a water bottle and a food dish.** B. Question: How do I ready a guinea pig cage for its new occupants? Answer: Provide the guinea pig with a cage full of a few inches of bedding made of ripped jeans material, you will also need to supply it with a water bottle and a food dish. *Training: cross-entropy over two scalar classification-head scores.*	`<|im_start|>user` Choose the most appropriate solution for the given goal. Goal: How do I ready a guinea pig cage for its new occupants? A. Provide the guinea pig with a cage full of a few inches of bedding made of ripped paper strips... B. Provide the guinea pig with a cage full of a few inches of bedding made of ripped jeans material... Answer with only one letter: A or B. Answer:`<|im_end|>` `<|im_start|>assistant` **A** *Training: next-token prediction on the answer label.*	*Only the gold solution is used:* Question: How do I ready a guinea pig cage for its new occupants? Answer: **Provide the guinea pig with a cage full of a few inches of bedding made of ripped paper strips, you will also need to supply it with a water bottle and a food dish.** *Training: next-token prediction only on the gold solution tokens; question/prefix tokens are masked.*

Table 9: Concrete examples of the three training paradigms. Blue text denotes shared context or question text, orange denotes incorrect candidate content, green denotes the gold candidate, and purple denotes the supervised answer label. CLS scores all candidates with a scalar classification head. Label-Gen SFT trains the model to emit a label token. Gold-Only SFT uses only the gold candidate sequence and applies next-token prediction only to the candidate/ending tokens.

B.2 Abstract Instruction Templates

Table 10 abstracts away from individual examples and shows the dataset-specific templates used to construct inputs. These templates are the actual formatting rules used to convert each supervised example into the three training objectives.

Dataset	CLS: candidate-sequence format	Label-Gen SFT: chat-template format	Gold-Only SFT: gold-only continuation format
HellaSwag	*Four candidates are constructed:* A. {ctx} {ending_0} B. {ctx} {ending_1} C. {ctx} {ending_2} D. {ctx} {ending_3} *Each candidate receives one scalar classification-head score; cross-entropy is applied over the four scores.*	`<|im_start|>user` Choose the most plausible ending for the given context. Context: {ctx} A. {ending_0} B. {ending_1} C. {ending_2} D. {ending_3} Answer with only one letter: A, B, C, or D. Answer:`<|im_end|>` `<|im_start|>assistant` **{A|B|C|D}** *Loss is applied only to the answer-label token.*	{ctx} **{gold ending}** *Only the correct ending is used. The context is masked, and next-token prediction loss is applied over the gold ending tokens.*
WinoGrande	*The sentence is split at the blank:* A. {prefix} {option_1} {suffix} B. {prefix} {option_2} {suffix} *Each substituted sentence receives one scalar classification-head score; cross-entropy is applied over the two scores.*	`<|im_start|>user` Fill in the blank in the following sentence with the most appropriate option. Sentence: {sentence with _ blank} A. {option_1} B. {option_2} Answer with only one letter: A or B. Answer:`<|im_end|>` `<|im_start|>assistant` **{A|B}** *Loss is applied only to the answer-label token.*	{prefix} **{gold option}** {suffix} *Only the correct substituted sentence is used. The prefix before the blank is masked, and next-token prediction loss is applied to the gold option and remaining suffix tokens.*
PIQA	*Two question-answer candidates are constructed:* A. Question: {goal}\nAnswer: {sol_1} B. Question: {goal}\nAnswer: {sol_2} *Each solution receives one scalar classification-head score; cross-entropy is applied over the two scores.*	`<|im_start|>user` Choose the most appropriate solution for the given goal. Goal: {goal} A. {sol_1} B. {sol_2} Answer with only one letter: A or B. Answer:`<|im_end|>` `<|im_start|>assistant` **{A|B}** *Loss is applied only to the answer-label token.*	Question: {goal}\nAnswer: **{gold solution}** *Only the correct solution is used. The question and* `Answer:` *prefix are masked, and next-token prediction loss is applied over the gold solution tokens.*

Table 10: Abstract input templates used by the three paradigms. CLS converts every answer option into a complete candidate sequence and trains a scalar scoring head with cross-entropy over candidates. Label-Gen SFT presents all choices in a chat-template prompt and supervises only the answer label. Gold-Only SFT uses the same candidate construction as CLS, but keeps only the gold candidate and trains the causal LM with next-token prediction over the gold answer/ending tokens.

C Detailed Evaluation Results

C.1 LoRA and Training Hyperparameters

All fine-tuning runs use Low-Rank Adaptation (LoRA) over all linear layers of the transformer backbone. We use rank $r = 16$, scaling factor $\alpha = 32$, and no bias adaptation. For the causal language-modeling baselines, LABELGEN and GOLDONLY, we use LoRA dropout 0.05. For CLS, we use the sequence-classification LoRA setup with dropout 0.0 (as we want each candidate to go through the same LoRA updates) and save the scalar classification score head using `modules_to_save = ["score"]`. Thus, the newly initialized classification head is trained in full together with the LoRA adapters, while the base model parameters remain frozen.

Setting	CLS	LABELGEN	GOLDONLY
Backbone model	Qwen3	Qwen3	Qwen3
HF model class	Seq. classification	Causal LM	Causal LM
LoRA task type	SEQ_CLS	CAUSAL_LM	CAUSAL_LM
Target modules	all linear	all linear	all linear
LoRA rank r	16	16	16
LoRA α	32	32	32
LoRA dropout	0.0	0.05	0.05
Bias adaptation	none	none	none
Saved extra modules	score head	–	–

Table 11: LoRA configuration used for the three fine-tuning paradigms. The CLS setting trains the scalar sequence-classification score head in addition to LoRA adapters. LABELGEN and GOLDONLY keep the causal language-modeling head and train only LoRA adapters.

Unless otherwise stated, all runs use the same optimizer and schedule: 2 epochs, learning rate 2×10^{-4}, per-device batch size 8, gradient accumulation over 8 steps, effective batch size 64, cosine learning-rate decay, warmup ratio 0.1 for CLS and 0.03 for the causal-LM baselines, weight decay 0.01, and gradient clipping at maximum norm 1.0.

C.2 Evaluation Protocol

Each trained model is evaluated in its native format. For CLS, we report the accuracy of the classification-head argmax over candidate scores. For Label-Gen SFT, we report next-token answer accuracy over the answer-letter tokens. For Gold-Only SFT, we evaluate candidate sequences using language-model log-probability scoring.

We additionally evaluate the Label-Gen SFT adapter as a candidate scorer by applying language-model log-probability scoring to the same candidate sequences used by CLS; we denote this evaluation as Label-Gen LogProb. This is not a separately trained model, but an apples-to-apples test of whether an instruction-tuned adapter also improves candidate ranking.

We report the following log-probability-based candidate-scoring metrics:

- **LP-Full:** log-probability of the full candidate sequence.
- **LP-Sum:** sum of suffix token log-probabilities conditioned on the context.
- **LP-Avg:** suffix log-probability normalized by suffix token count.
- **LP-Char:** suffix log-probability normalized by suffix character length.

This appendix reports the full set of evaluation metrics used in the paper. The main text focuses on the native accuracy of the trained objective, while the tables below also include language-model log-probability based re-scoring variants. These additional columns help separate improvements in answer-label generation from improvements in candidate-sequence ranking.

C.3 Commonsense Benchmarks

Table 12 gives the full results on the three primary commonsense benchmarks. For each dataset and model size, we compare the proposed classification-head objective against instruction-style supervised fine-tuning and continuation language-model fine-tuning.

Dataset	Model	Stage	Trained	LP-Full	LP-Sum	LP-Avg	LP-Char	McNemar p †
HellaSwag	Qwen3-0.6B	CLS	**80.95**	36.39	36.36	47.76	48.19	2.8×10^{-11}**
		Label-Gen SFT	78.50	60.07	64.10	57.52	56.29	
		Label-Gen LogProb	—	36.54	36.37	45.97	47.93	
		Gold-Only SFT	—	40.75	40.90	50.99	52.98	
	Qwen3-1.7B	CLS	**90.81**	43.40	43.52	57.86	59.15	8.5×10^{-8}**
		Label-Gen SFT	89.33	80.48	82.29	80.94	80.42	
		Label-Gen LogProb	—	45.01	45.14	57.81	59.59	
		Gold-Only SFT	—	50.09	50.10	64.71	66.75	
	Qwen3-4B	CLS	**94.24**	46.23	46.32	63.94	65.46	0.105 n.s.
		Label-Gen SFT	93.90	91.10	92.94	91.73	91.29	
		Label-Gen LogProb	—	50.76	50.75	65.57	68.29	
		Gold-Only SFT	—	57.50	57.49	73.32	75.48	
	Qwen3-8B	CLS	**95.62**	54.12	54.12	73.72	75.23	0.598 n.s.
		Label-Gen SFT	95.51	95.25	95.25	90.27	88.36	
		Label-Gen LogProb	—	55.51	55.51	72.37	74.81	
		Gold-Only SFT	—	61.65	61.65	78.08	80.15	
WinoGrande	Qwen3-0.6B	CLS	**70.80**	62.37	61.80	61.40	—	0.0058**
		Label-Gen SFT	67.80	67.12	65.65	65.48	—	
		Label-Gen LogProb	—	57.56	57.50	56.82	—	
		Gold-Only SFT	—	57.61	58.23	57.61	—	
	Qwen3-1.7B	CLS	**78.16**	67.01	67.01	66.16	—	0.518 n.s.
		Label-Gen SFT	77.48	76.51	76.00	75.95	—	
		Label-Gen LogProb	—	63.38	62.93	62.37	—	
		Gold-Only SFT	—	66.33	66.33	65.53	—	
	Qwen3-4B	CLS	**87.21**	73.34	73.34	72.33	—	1.000 n.s.
		Label-Gen SFT	87.15	84.10	83.98	84.38	—	
		Label-Gen LogProb	—	67.23	67.52	66.67	—	
		Gold-Only SFT	—	68.42	68.59	67.74	—	
	Qwen3-8B	CLS	**89.53**	73.85	73.63	72.50	—	0.733 n.s.
		Label-Gen SFT	89.25	84.15	80.59	80.25	—	
		Label-Gen LogProb	—	69.84	69.84	69.89	—	
		Gold-Only SFT	—	71.99	72.78	72.21	—	
PIQA	Qwen3-0.6B	CLS	**77.80**	65.56	65.23	65.56	66.27	0.0034**
		Label-Gen SFT	74.70	73.83	73.01	72.63	72.63	
		Label-Gen LogProb	—	67.41	67.41	67.19	68.34	
		Gold-Only SFT	—	72.03	72.03	71.33	71.87	
	Qwen3-1.7B	CLS	**85.04**	69.26	69.37	69.75	70.24	0.0060**
		Label-Gen SFT	82.59	59.14	57.89	58.54	58.16	
		Label-Gen LogProb	—	73.07	73.45	74.27	74.86	
		Gold-Only SFT	—	76.93	76.82	77.26	77.48	
	Qwen3-4B	CLS	**89.66**	74.65	74.86	75.30	76.77	0.754 n.s.
		Label-Gen SFT	89.39	88.30	88.90	89.12	88.79	
		Label-Gen LogProb	—	76.06	75.79	76.33	76.93	
		Gold-Only SFT	—	79.22	78.99	80.09	79.98	
	Qwen3-8B	CLS	91.19	75.79	75.95	76.88	77.15	0.201 n.s.
		Label-Gen SFT	**92.11**	91.13	90.37	88.47	87.87	
		Label-Gen LogProb	—	78.45	78.40	79.82	79.92	
		Gold-Only SFT	—	81.56	81.45	82.32	82.05	

Table 12: Test-set accuracy (%) for the three commonsense benchmarks under all four evaluated stages. TRAINED reports CLS head accuracy for Stage 1 rows and NEXTTOK accuracy for Stage 2 rows. Label-Gen LogProb re-scores the Stage 2 SFT adapter using the CLS log-prob format for an apples-to-apples comparison. LP variants are defined in Appendix C.2. **Bold** marks the highest accuracy per (dataset, model) block. † McNemar exact two-sided binomial p-value comparing CLS and Label-Gen SFT on per-example correctness; $^{**}p < 0.01$, $^{*}p < 0.05$. All rows use the unified setup of Appendix C.1.

C.4 Log-Probability Normalization Effects

The log-probability evaluations reveal how sensitive candidate ranking is to sequence-length normalization. Across HellaSwag and PIQA, LP-Char is often stronger than LP-Avg for the candidate-sequence evaluations most relevant to our analysis, namely CLS and Label-Gen LogProb. This suggests that character-level normalization can better control for answer-length effects in these datasets. The pattern is not universal: native Label-Gen SFT log-probability scores often prefer LP-Avg, and WinoGrande does not use LP-Char in our evaluation.

SFT does not necessarily transfer to candidate scoring. Label-Gen LogProb is consistently lower than native Label-Gen SFT accuracy, even though both use the same trained adapter. This indicates that Label-Gen SFT primarily improves the model's ability to emit the correct answer label in the chat-template format, but does not reliably improve its ability to rank the candidate answer sequences themselves. Gold-Only SFT is usually stronger than Label-Gen LogProb under normalized log-probability scoring, but remains far below CLS. This suggests that increasing the likelihood of the gold continuation alone helps candidate scoring somewhat, but is weaker than directly training a discriminative objective over correct and incorrect candidates.

C.5 Science Benchmarks

Table 13 reports the same evaluation suite on ARC-Challenge and SciQ. These benchmarks are used as additional science-domain evaluations rather than the primary benchmark suite.

Dataset	Model	Stage	Trained	LP-Full	LP-Sum	LP-Avg	LP-Char	McNemar p †
ARC-Challenge	Qwen3-0.6B	CLS	63.61	36.57	36.48	40.17	39.31	0.817 n.s.
		Label-Gen SFT	**64.03**	63.00	63.52	63.78	64.03	
		Label-Gen LogProb	—	34.25	34.68	37.17	36.39	
		Gold-Only SFT	—	39.23	39.74	44.21	42.75	
	Qwen3-1.7B	CLS	79.57	52.70	52.70	55.11	56.31	1.000 n.s.
		Label-Gen SFT	**79.66**	78.80	79.48	79.40	79.57	
		Label-Gen LogProb	—	39.66	39.83	42.40	43.09	
		Gold-Only SFT	—	51.59	51.33	56.48	55.71	
	Qwen3-4B	CLS	88.33	53.65	53.73	56.14	58.11	0.178 n.s.
		Label-Gen SFT	**89.70**	89.53	89.79	89.79	89.70	
		Label-Gen LogProb	—	52.10	52.53	54.68	56.31	
		Gold-Only SFT	—	59.91	60.09	65.06	64.03	
	Qwen3-8B	CLS	91.07	61.80	61.97	64.64	65.15	0.121 n.s.
		Label-Gen SFT	**92.45**	92.88	92.45	92.19	91.93	
		Label-Gen LogProb	—	56.65	56.31	57.85	58.88	
		Gold-Only SFT	—	64.55	64.64	68.50	68.84	
SciQ	Qwen3-0.6B	CLS	**90.70**	68.70	68.30	72.80	70.50	0.0263*
		Label-Gen SFT	88.50	86.80	87.10	86.20	84.60	
		Label-Gen LogProb	—	76.20	76.50	75.60	75.60	
		Gold-Only SFT	—	84.60	85.10	85.50	86.00	
	Qwen3-1.7B	CLS	**94.20**	83.40	83.80	85.10	84.90	0.203 n.s.
		Label-Gen SFT	93.20	91.80	91.20	90.30	89.10	
		Label-Gen LogProb	—	83.50	83.40	84.20	82.80	
		Gold-Only SFT	—	91.50	91.30	92.00	92.40	
	Qwen3-4B	CLS	95.00	89.40	89.70	90.70	90.80	1.000 n.s.
		Label-Gen SFT	**95.10**	95.20	95.20	95.30	95.20	
		Label-Gen LogProb	—	90.60	90.70	90.10	90.80	
		Gold-Only SFT	—	92.60	92.90	92.70	93.00	
	Qwen3-8B	CLS	96.00	93.20	93.40	93.50	94.00	0.188 n.s.
		Label-Gen SFT	**96.90**	97.00	96.70	96.80	96.40	
		Label-Gen LogProb	—	92.20	92.60	92.50	92.20	
		Gold-Only SFT	—	94.80	94.50	94.70	95.80	

Table 13: Test-set accuracy (%) on the two science benchmarks ARC-Challenge and SciQ. Format and metric definitions are identical to Table 12. **Bold** marks the highest accuracy per (dataset, model) block. Unlike the commonsense benchmarks, the science benchmarks do not show a consistent advantage for CLS: Label-Gen SFT directionally wins on all ARC-Challenge settings and on the larger SciQ models, while CLS is significantly better only on SciQ at 0.6B.