Supplementary Material for Accelerating On-Device LLM Inference via Activation-guided FFN Distillation on Raspberry Pi

Table 1. CoQA performance under different phase-selective modes (LLaMA-3.2-1B, PyTorch FP16). Under equal token counts (256 tokens each), the prefill and decode phases account for approximately 13% and 87% of total inference time, respectively. Note that † denotes our baseline configuration.

Mode	FFN at Prefill	FFN at Decode	CoQA	
			EM	F1
Baseline	Baseline	Baseline	0.539	0.695
Prefill-AFD	AFD	Baseline	0.453	0.573
Decode-AFD†	Baseline	AFD	0.536	0.692
Both-AFD	AFD	AFD	0.395	0.510

1. Impact of Phase-Selective AFD

Table 1 compares CoQA performance across different phase-selective AFD configurations, where the prefill and decode phases account for 13% and 87% of total inference time, respectively, confirming that the decode phase is the dominant computational bottleneck.

Applying AFD exclusively to the decode phase (Decode-AFD) nearly matches the baseline in both EM and F1, whereas applying AFD to the prefill phase results in a substantial performance drop. Compressing both phases leads to the most severe degradation, indicating that distributional shift compounds across phases.

These results confirm that the prefill phase is critical for establishing contextual representation, while the decode phase can be effectively compressed without significant quality loss. Therefore, applying AFD exclusively to the decode phase achieves the optimal trade-off between computational efficiency and generation fidelity.

2. Effect of Calibration Dataset Size

Table 2 shows the effect of calibration dataset size on AFD performance. Performance consistently improves as the number of calibration sequences increases from 128 to 1,024, with CoQA EM rising from 0.257 to 0.536 and F1 from 0.421 to 0.692. However, increasing the dataset size beyond 1,024 sequences results in slight performance degradation (EM: 0.521, F1: 0.675), suggesting that overfitting to the calibration distribution occurs beyond a certain threshold. Based on these results, we set the default calibration set size to 1,024 sequences, as this achieves the best trade-off between data efficiency and task performance.

Table 2. Effect of calibration dataset size on AFD performance (CoQA). Benchmark performance is computed on the PyTorch format with FP16 precision. † denotes our baseline configuration. Red indicates the best performance within the above models.

# Calib. Sequences	CoQA	
	EM	F1
128	0.257	0.421
256	0.425	0.551
512	0.503	0.623
1024†	0.536	0.692
2048	0.521	0.675

Table 3. Inference speed comparison across different FFN intermediate dimensions ($\hat{d}_{\text{ff}}$) on Raspberry Pi 4 with LLaMA-3.2-1B. Improvement ratios are calculated relative to the baseline ($d_{\text{ff}} = 8192$).

Format	Method	$\hat{d}_{\text{ff}} = 2048$	$\hat{d}_{\text{ff}} = 4096$	$\hat{d}_{\text{ff}} = 6144$
PyTorch (FP16)	Baseline (8192)	1.29	1.29	1.29
	AFD (Ours)	1.60	1.36	1.18
	Improvement	+24.0%	+5.4%	-8.5%

3. Runtime Dependency Limitations

While structural compression theoretically implies a linear reduction in computational overhead, our empirical results on the PyTorch runtime (Table 3) reveal a non-linear relationship between FFN dimensions and actual inference speed. Specifically, although reducing the intermediate dimension to $\hat{d}_{\text{ff}} = 2048$ and 4096 yields speedups of 24.0% and 5.4% over the baseline respectively, the improvement does not scale proportionally with the reduction in parameters. More importantly, at $\hat{d}_{\text{ff}} = 6144$, the inference speed (1.18 tokens/s) actually degrades by 8.5% compared to the baseline (1.29 tokens/s). These results indicate that the efficacy of AFD is heavily constrained by the underlying run-

time's memory management and kernel optimization strategies, particularly when the reduced FFN dimensions do not align with the optimal memory tiling sizes or SIMD vectorization patterns of the target runtime. This runtime dependency remains a significant limitation, and future work should explore co-design between compression strategies and low-level kernel optimizations to ensure consistent and predictable inference gains across diverse hardware backends.

4. Benchmark Performance on Other Architectures

To further evaluate the generalizability of our proposed AFD across different model architectures, we conducted additional experiments using the Gemma-3-1B model.

As detailed in Table 4, kernel-level profiling of Gemma-3-1B reveals that the FFN constitutes a significant computational bottleneck. Specifically, under the PyTorch FP16 environment, the FFN accounts for 45.2% (460.3 ms) of the total inference time.

Table 4. Module-wise Profiling (execution time with ratio) for Gemma-3-1B. Experiments capture the latency breakdown under the PyTorch (FP16) configuration.

Precision	Attn	FFN	LM Head	Others
FP16	248.2 ms (24.4%)	460.3 ms (45.2%)	207.8 ms (20.4%)	101.8 ms (10.0%)
FP16 (AFD)	248.6 ms (30.9%)	247.9 ms (30.8%)	209.0 ms (26.0%)	98.9 ms (12.3%)

By applying AFD, this structural inefficiency is effectively mitigated. Consistent with the trends observed in the LLaMA architecture, AFD significantly accelerates the FFN module in Gemma-3-1B, reducing its execution time from 460.3 ms to 247.9 ms. Notably, the latencies of the attention mechanism and LM head remain largely unaffected, confirming that AFD precisely targets the FFN bottleneck without introducing unintended framework overhead.

As shown in Table 5, this module-level acceleration translates into a notable increase in overall decoding speed, improving from 1.21 TPS to 1.44 TPS (a 19.0% speedup). Crucially, this acceleration is achieved with only a marginal degradation in generation quality, where the CoQA EM and F1 scores exhibit minimal drops.

Consequently, the overall challenge score improves substantially from 1.845 to 2.178. These results confirm that the massive FFN computational overhead is a shared structural characteristic among modern LLMs, and demonstrate that our proposed AFD method reliably accelerates distinct LLM architectures.

5. Response Quality Assessment and Limitations

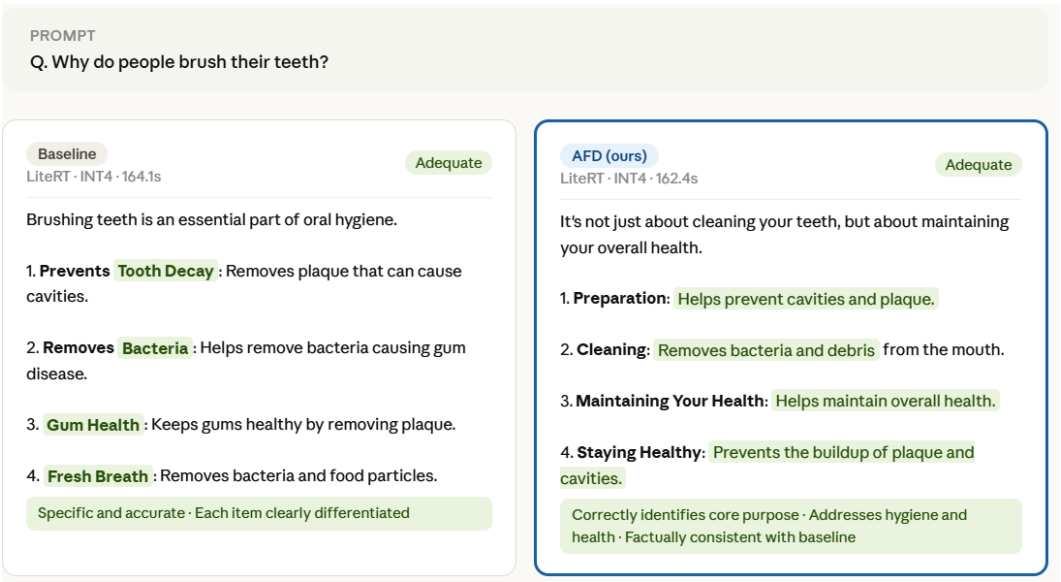
Figure 1a presents a qualitative comparison on an open-ended question, where the model is asked to explain why people brush their teeth without any provided context. Both the baseline and AFD successfully generate factually relevant responses that correctly identify the core purpose of brushing teeth, including cavity prevention, bacteria removal, and overall oral hygiene. Although AFD's response is less specific than the baseline in terms of item differentiation, it nonetheless demonstrates a correct understanding of the topic and produces a coherent, factually consistent answer. These results suggest that AFD preserves the model's ability to generate appropriate responses on open-ended knowledge-based questions even after FFN compression.

Table 5. Benchmark Performance of AFD on Gemma-3-1B. Note that the Challenge Score is computed using the average of the CoQA EM and F1 metrics.

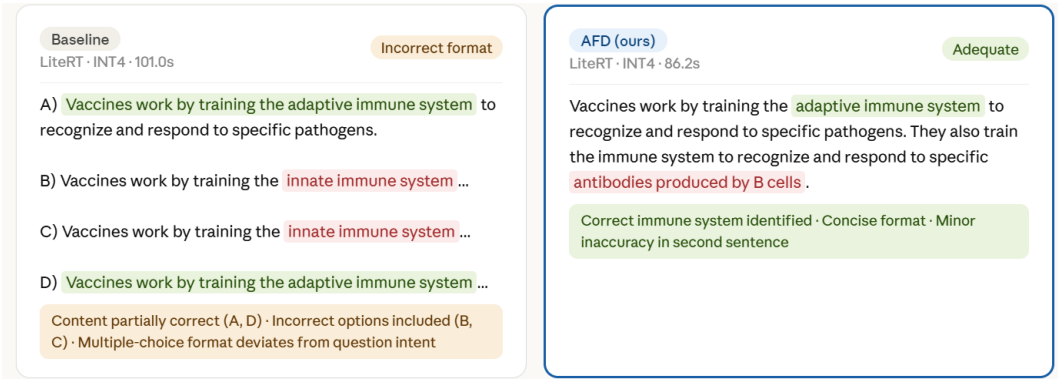
Precision	Method	Speed	CoQA		Score
		(TPS)	EM	F1	
FP16	Baseline	1.21	0.539	0.678	1.845
	AFD (ours)	1.44	0.525	0.669	2.178

Figure 1b presents a qualitative comparison on a context-based question, where the model is required to identify how vaccines work and which part of the immune system they train. AFD produces a concise and direct response, correctly identifying the *adaptive immune system* as the target of vaccine training, which is consistent with the ground truth answer derived from the passage. In contrast, the baseline generates an unnecessary multiple-choice format that includes incorrect options (B and C), which erroneously refer to the *innate immune system*, despite the correct answer being present in options A and D. These results demonstrate that AFD preserves the ability to extract and articulate context-grounded factual information in a coherent and appropriately formatted response, suggesting that the proposed compression does not significantly impair the model's instruction-following and comprehension capabilities.

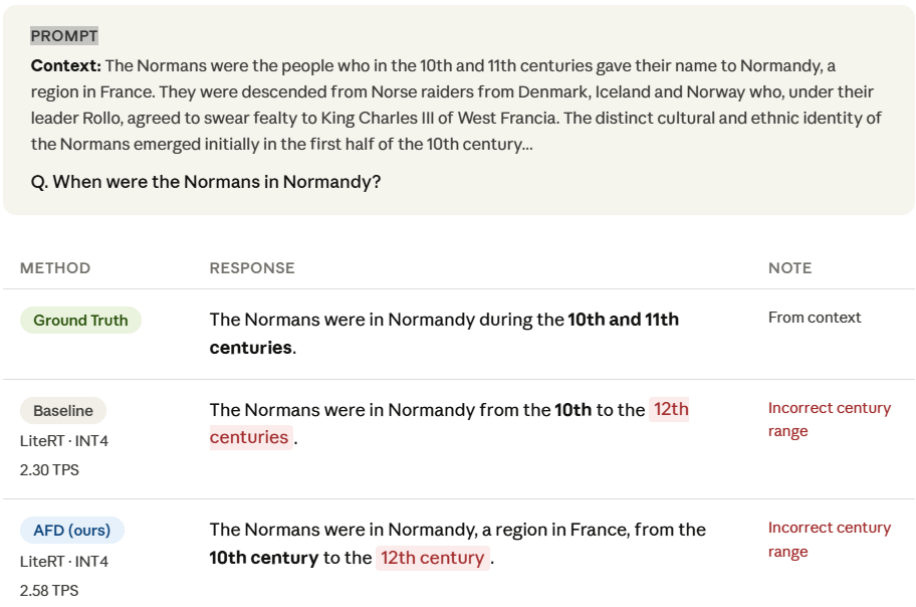
Despite the inference gains achieved by AFD, the proposed method inherits the generation limitations of the underlying baseline model. As illustrated in Figure 1c, both the baseline and AFD produce factually incorrect responses, extending the temporal range of the Normans in Normandy to the 12th century, whereas the ground truth is confined to the 10th and 11th centuries. This suggests that AFD, which compresses and distills the FFN sub-layers without modifying the attention mechanism or the model's parametric knowledge, cannot correct errors that are already present in the baseline model. Consequently, the quality of the generated output remains fundamentally bounded by the capability of the original model, and factual hallucinations inherited from the baseline are preserved after compression.



(a) Similar case of open-ended question.



(b) Similar case of context-based question.



(c) Limitation case of context-based question.

Figure 1. Response quality comparison between the INT4 LiteRT Baseline and AFD (ours).