Supplementary Material for LiteRT-Optimized INT8 LLM for Raspberry Pi4 Deployment

A. Efficiency Evaluation on Other Target Platforms

To expand the scope of our evaluation beyond the Raspberry Pi 4, we conducted additional experiments to assess the inference efficiency of various LLMs on distinct type of hardware platforms: a high-end server-class CPU and the Raspberry Pi 5. Specifically, we measured the inference throughput in terms of Token Per Second (TPS), evaluating performance across short and long input sequences, as well as their average. For the server-class CPU, we utilized an AMD Ryzen 9 7950X processor equipped with 128GB of RAM. On the edge device, we used the Raspberry Pi 5 for benchmarking, as it provides a more realistic assessment of low-power, resource-constrained deployment scenarios.

As shown in the main paper, we selected five representative LLMs with relatively small parameters and varying architectures. Each model was evaluated using three different execution formats: FP16 and INT8 precision under the standard PyTorch runtime, and INT8 execution through LiteRT—an optimized inference framework designed for high-throughput, low-latency CPU execution.

Table 1 presents the detailed TPS results. Across all models and platforms, with the exception of the QWEN model, the LiteRT backend consistently outperformed both FP16 and PyTorch-based INT8 inference. This performance advantage was particularly evident on the Raspberry Pi 5, where LiteRT delivered 2× to 3× higher throughput than PyTorch INT8 in most configurations. For example, the Llama-3.2 1B model achieved 5.08 TPS using LiteRT, compared to just 2.42 TPS using PyTorch INT8.

Similarly, larger models like DeepSeek-R1-Distill-Qwen and compact architectures like TinyLlama showed substantial performance gains when executed via LiteRT. TinyLlama reached up to 6.55 TPS on Raspberry Pi 5 under LiteRT, nearly doubling its PyTorch INT8 performance. These findings demonstrate not only the scalability of LiteRT across a range of model sizes, but also its portability and effectiveness in optimizing inference for heterogeneous CPU platforms, from edge devices to desktop-class processors.

B. Additional Runtime Evaluation on Mobile SoCs

To further assess the practicality of deploying large language models (LLMs) on consumer devices, we conducted a comprehensive runtime evaluation across several representative mobile System-on-Chip (SoC) platforms. Specifically, we benchmarked INT8 quantized versions of five models using the LiteRT runtime on four mobile devices: Galaxy S24 Ultra (Snapdragon Gen3), Galaxy Fold 4 (Snapdragon Gen1), Galaxy S24 (Exynos 2400), and Galaxy Tab S7+ (Snapdragon 865). To ensure consistent evaluation across all platforms, all tests were performed using the CPU configuration in the official Google AI Edge Gallery application [1].

Table 2 demonstrated the TPS for each configuration. The results reveal clear performance scaling with hardware capability. Devices with more recent chipsets, such as the Snapdragon Gen3 and Exynos 2400, exhibit significantly higher TPS compared to older chips like the Snapdragon 865. For example, Qwen3 (0.6B) reaches an impressive 36.8 TPS on the S24 Ultra, indicating that LiteRT is highly optimized to exploit modern CPU architectures without relying on GPU or NPU acceleration.

We also observe that compact and well-optimized models like TinyLlama (1.1B) and Qwen3 (0.6B) outperform heavier counterparts like Llama-3.2 3B, particularly on resource-constrained hardware. For instance, Llama-3.2 3B yields only 0.32 TPS on the Exynos 2400-based S24, highlighting the practical limitations of deploying large-scale models on mobile CPUs. In contrast, TinyLlama achieves 23.5 TPS on the same device—over 70× faster.

Furthermore, LiteRT's ability to maintain high throughput with compact models such as Qwen3 and TinyLlama, demonstrates its effectiveness in leveraging model efficiency for devices with limited resources.

Overall, these findings reinforce the versatility and scalability of LiteRT across a diverse set of mobile SoCs. They also emphasize the critical importance of model efficiency for achieving real-time LLM inference on handheld and embedded devices without requiring specialized acceleration

[1] https://github.com/google-ai-edge/gallery

model	Params	format	precision	Token Per Second (TPS)					
				Server			Raspbery Pi 5		
				short	long	avg	short	long	avg
Llama-3.2 [1]	1B	PyTorch	FP16	3.44	3.11	3.28	3.21	2.93	3.07
		PyTorch	INT8	21.11	21.67	21.39	2.48	2.36	2.42
		LiteRT	INT8	23.45	23.47	23.46	5.11	5.05	5.08
Llama-3.2 [1]	3B	PyTorch	FP16	0.96	0.80	0.88	1.23	1.15	1.19
		PyTorch	INT8	9.65	9.72	9.69	-	-	-
		LiteRT	INT8	7.66	7.56	7.67	1.53	1.49	1.51
Qwen3 [32]	0.6B	PyTorch	FP16	4.02	4.16	4.09	3.68	3.76	3.72
		PyTorch	INT8	23.14	23.38	23.26	2.83	2.81	2.82
		LiteRT	INT8	13.82	13.78	13.80	2.87	2.83	2.85
DeepSeek-R1-Distill-Qwen [12]	1.5B	PyTorch	FP16	2.94	2.64	2.80	1.91	1.85	1.88
		PyTorch	INT8	16.60	16.39	16.50	1.88	1.81	1.85
		LiteRT	INT8	20.15	19.88	20.02	4.67	4.77	4.72
TinyLlama [36]	1.1B	PyTorch	FP16	5.17	5.05	5.11	2.69	2.73	2.71
		PyTorch	INT8	24.26	23.32	23.79	3.97	3.92	3.95
		LiteRT	INT8	28.49	28.06	28.28	6.62	6.48	6.55

Table 1. Latency or throughput measurements for different LLMs across formats and precision on server-class CPU and Rasberry Pi 5.

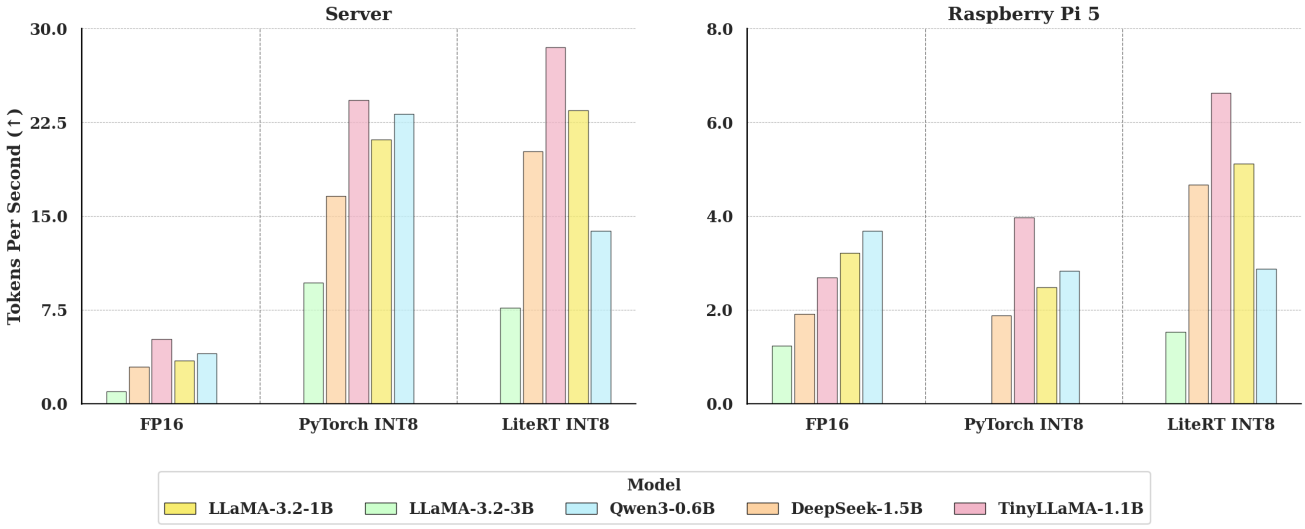

Figure 1. Compare TPS between quantized methods and LLMs on server-class CPU and Raspberry Pi 5.

hardware. As edge AI becomes more popular, the portability and extensibility of LiteRT will be essential for enabling real-time LLMs applications.

C. Comparison with Llama.cpp on Raspberry Pi

To better contextualize LiteRT's performance in edge environments, we conducted a direct comparison with llama.cpp, a widely adopted framework for running large language models efficiently on CPUs without requiring GPU or accelerator support. Table 3 summarizes the results of this comparison across five representative models, all quantized to INT8 and executed on both Raspberry Pi 4 and Raspberry Pi 5.

Across the configuration on Raspberry Pi, llama.cpp demonstrates throughput that is competitive with LiteRT.

model	Params	Token Per Second (TPS)			
		S24 Ultra (Snapdragon Gen3)	Fold 4 (Snapdragon Gen1)	S24 (Exynos 2400)	Tab S7+ (Snapdragon 865)
		avg	avg	avg	avg
Llama-3.2 [1]	1B	25.57	17.89	15.7	9.91
Llama-3.2 [1]	3B	8.82	6.85	0.32	2.84
Qwen3 [32]	0.6B	36.83	27.79	20.26	12.15
DeepSeek-R1-Distill-Qwen [12]	1.5B	21.28	16.23	13.48	8.08
TinyLlama [36]	1.1B	32.14	23.02	23.52	11.00

Table 2. LiteRT latency or throughput measurements for different LLMs on mobile devices.

model	Params	format	Precision	Model Size (GB)	Token Per Second (TPS)	
					Raspberry Pi 4	Raspberry Pi 5
Llama-3.2 [1]	1B	Llama.cpp	INT8	1.22	2.72	6.86
Llama-3.2 [1]	3B			3.18	1.05	2.6
Qwen3 [32]	0.6B			0.76	5.13	13.9
DeepSeek-R1-Distill-Qwen [12]	1.5B			1.76	2.17	5.59
TinyLlama [36]	1.1B			1.09	3.19	8.13

Table 3. Llama.cpp latency or throughput measurements for different LLMs on Raspberry Pi 4 and Raspberry Pi 5.

For instance, on Raspberry Pi 5, the Qwen3 model achieves 13.9 TPS using llama.cpp, outperforming most typical LiteRT configurations for models of similar size.

Unlike llama.cpp, which primarily targets desktop and terminal environments, LiteRT supports TensorFlow Lite export, making it compatible with Android and embedded platforms through native mobile apps. This makes LiteRT not only a high-performance runtime, but also a more general-purpose solution for production deployment across diverse hardware and software stacks.