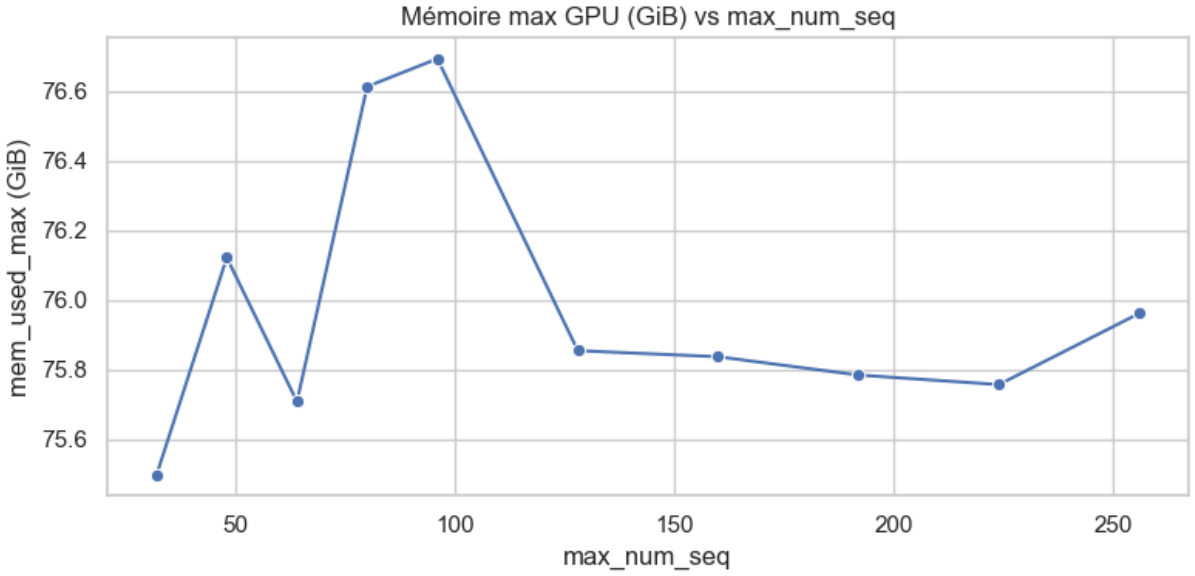
GPT optimisation

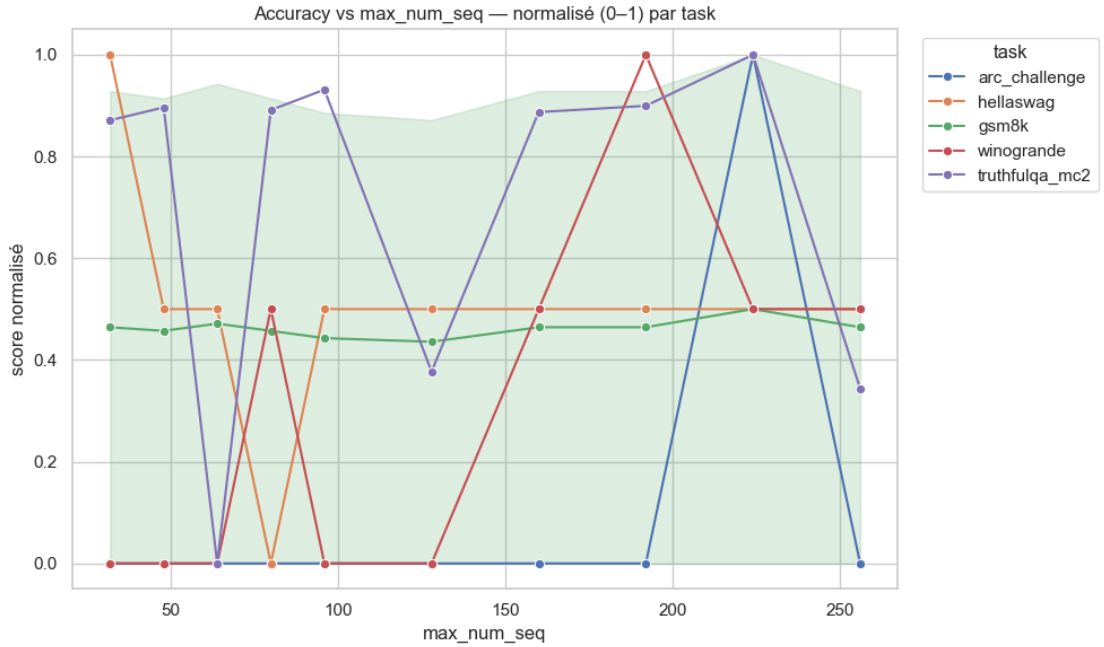
1. Max_num_seq

Figure 1) Max GPU memory (GiB) vs max_num_seq



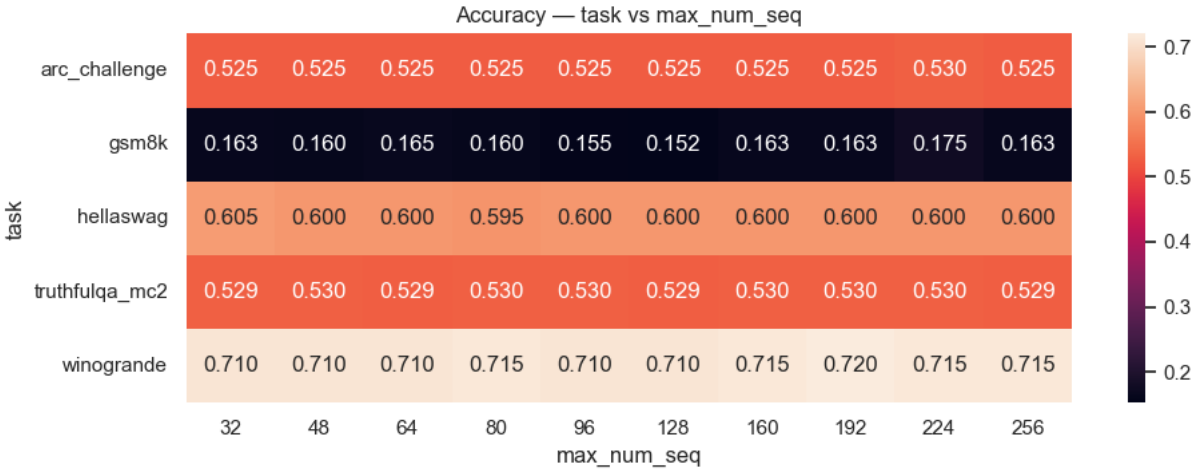
This plot shows that GPU memory usage is not monotonic with max_num_seq. Memory increases noticeably in the mid-range (around 80–96), reaching the highest observed values, then decreases and stabilizes again for larger values (roughly 128–224). Importantly, max_num_seq = 192 lies on a low and stable memory plateau, close to the minimum values observed in the sweep. This indicates that 192 provides good memory safety (lower VRAM pressure) compared with the peak region.

Figure 2) Accuracy vs max_num_seq (normalized 0–1 per task)



After normalizing each task to [0–1], the curves reveal that most tasks are fairly stable across the sweep, with only small variations. We see a few local peaks (some tasks improve around higher values), but these changes remain limited. max_num_seq = 192 consistently stays in the upper range of normalized performance across tasks, indicating it is a reliable choice rather than a fragile "one-task peak" configuration.

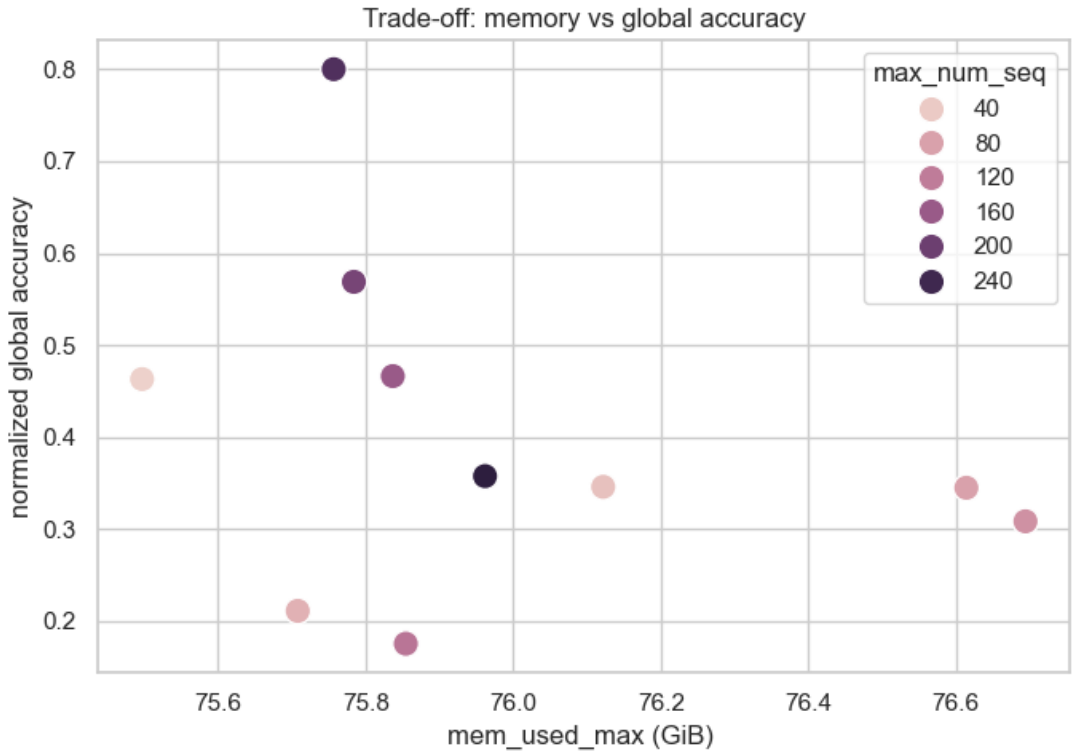
Figure 3) Heatmap of per-task accuracy vs max_num_seq



The heatmap gives the clearest "at a glance" view: accuracy values are generally very close across settings, confirming that max_num_seq does not drastically impact accuracy for this benchmark setup. A few cells show small improvements at certain values (for example, some tasks slightly increase at higher max_num_seq), but the overall pattern is flat. max_num_seq = 192 is among the best-balanced points: it

achieves strong results (notably high Winogrande) while remaining comparable to the best settings on the other tasks

Figure 4) Trade-off: memory vs global accuracy (normalized)



This scatter plot summarizes the decision problem directly: we want high accuracy with low memory. The key takeaway is that the best trade-offs are located in the lower-memory region, not in the memory peak region (80–96). max_num_seq = 192 appears in a favorable zone: it combines relatively low memory with one of the higher global-accuracy values, making it a strong compromise point.

Conclusion

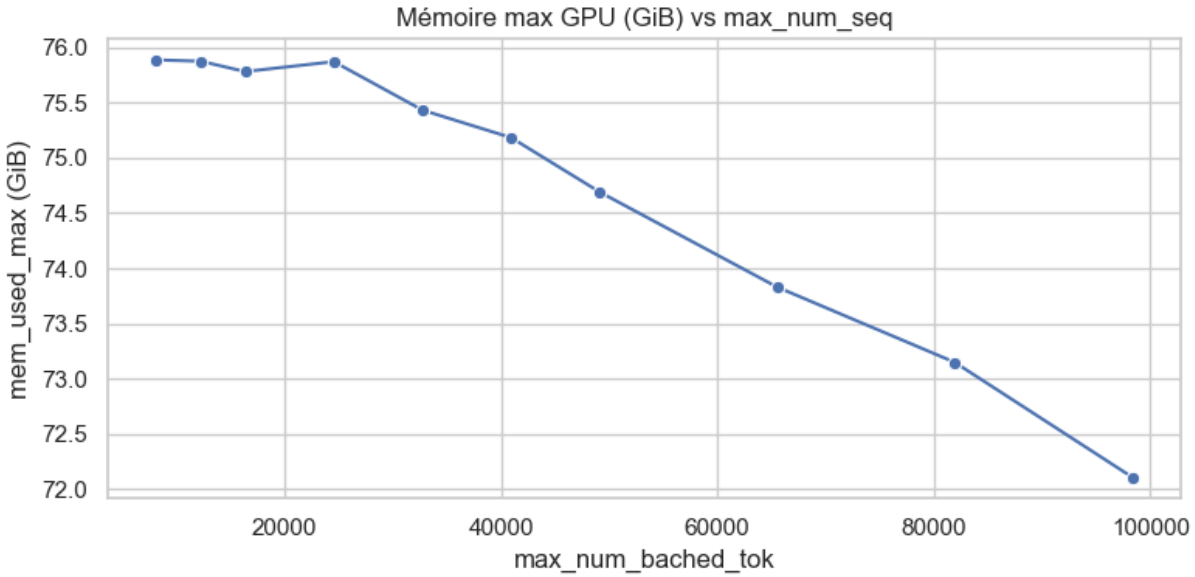
Across the sweep, GPU memory shows meaningful variation with max_num_seq, while accuracy remains largely stable with only minor fluctuations. The memory curve highlights a clear “high-pressure” region around 80–96, whereas values in the 128–224 range fall into a lower and more stable memory regime. Considering the per-task accuracy trends and the global memory–accuracy trade-off, we select **max_num_seq = 192** as the best operating point for this phase: it sits on the low-memory plateau and delivers consistently strong accuracy without relying on narrow, potentially noisy peaks.

2. Max_num_bached_tok

Based on the four plots for the max_num_batched_tok sweep, the main takeaway is that accuracy remains essentially stable, while GPU memory changes strongly and

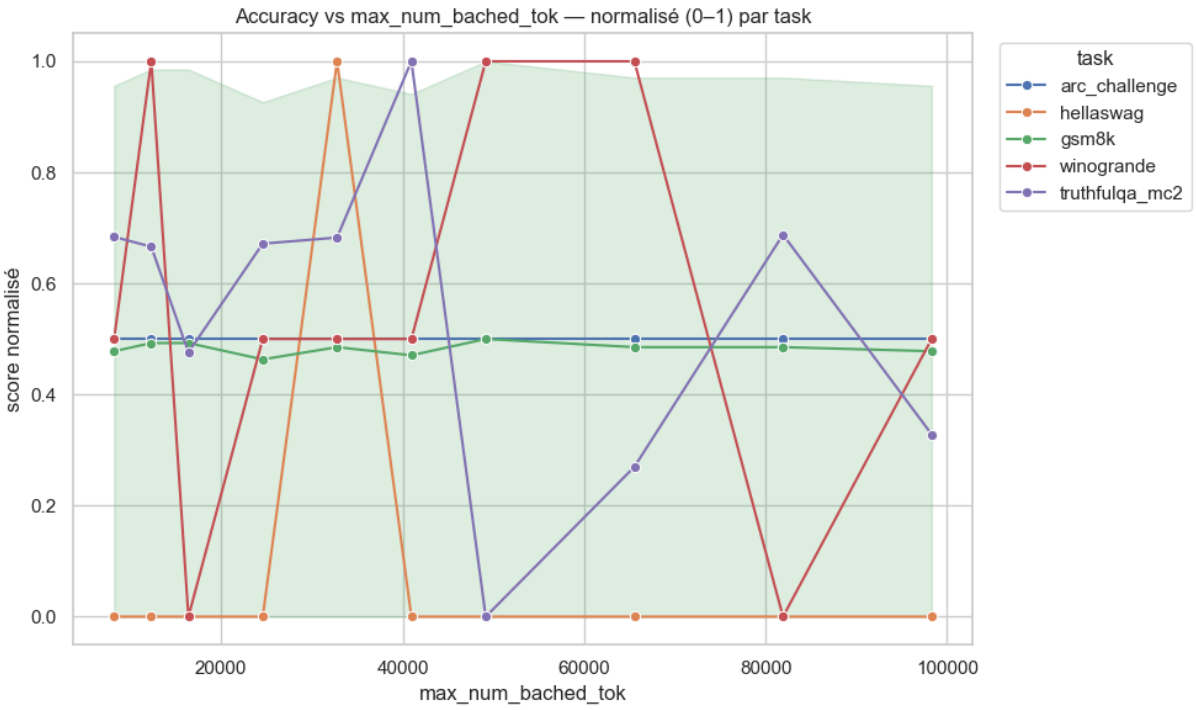
consistently. This makes the decision largely driven by the memory profile, with accuracy used as a constraint.

Figure 1) Max GPU memory vs max_num_batched_tok



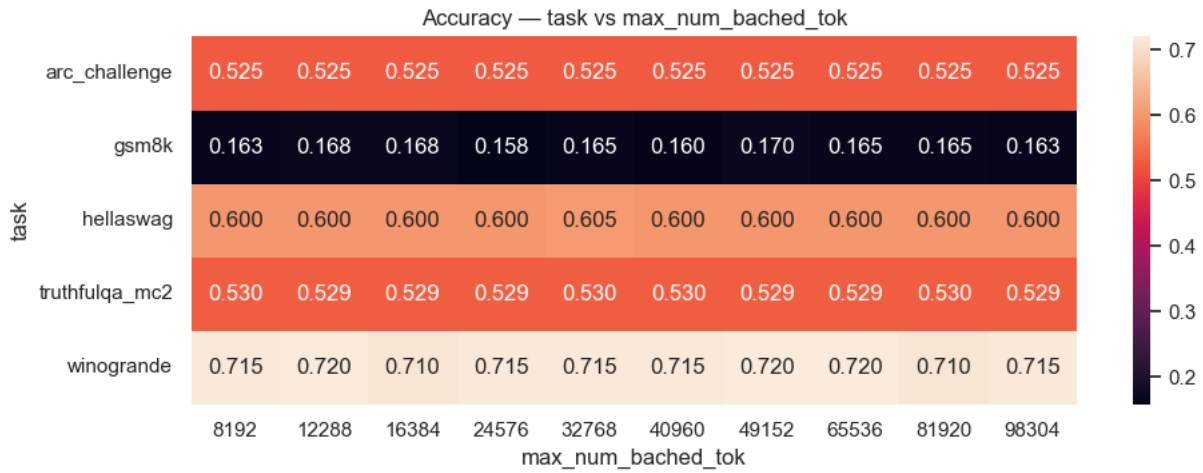
The memory curve shows a clear and almost monotonic improvement: increasing max_num_batched_tok steadily reduces peak VRAM (from ~76 GiB down to ~72 GiB). This is the strongest signal in the experiment. It means that higher values provide more headroom and reduce the risk of memory pressure, especially important when scaling load, increasing context length, or adding concurrency.

Figure 2) Normalized accuracy curves (per task)



The normalized (0–1) curves show visible spikes, but these are largely an artifact of normalization: most tasks vary only by a few thousandths, so min–max scaling can make tiny differences look large. This plot is still useful to see that there is no systematic accuracy collapse at high max_num_batched_tok, but it should not be used alone to pick an "optimal" value.

3) Accuracy heatmap



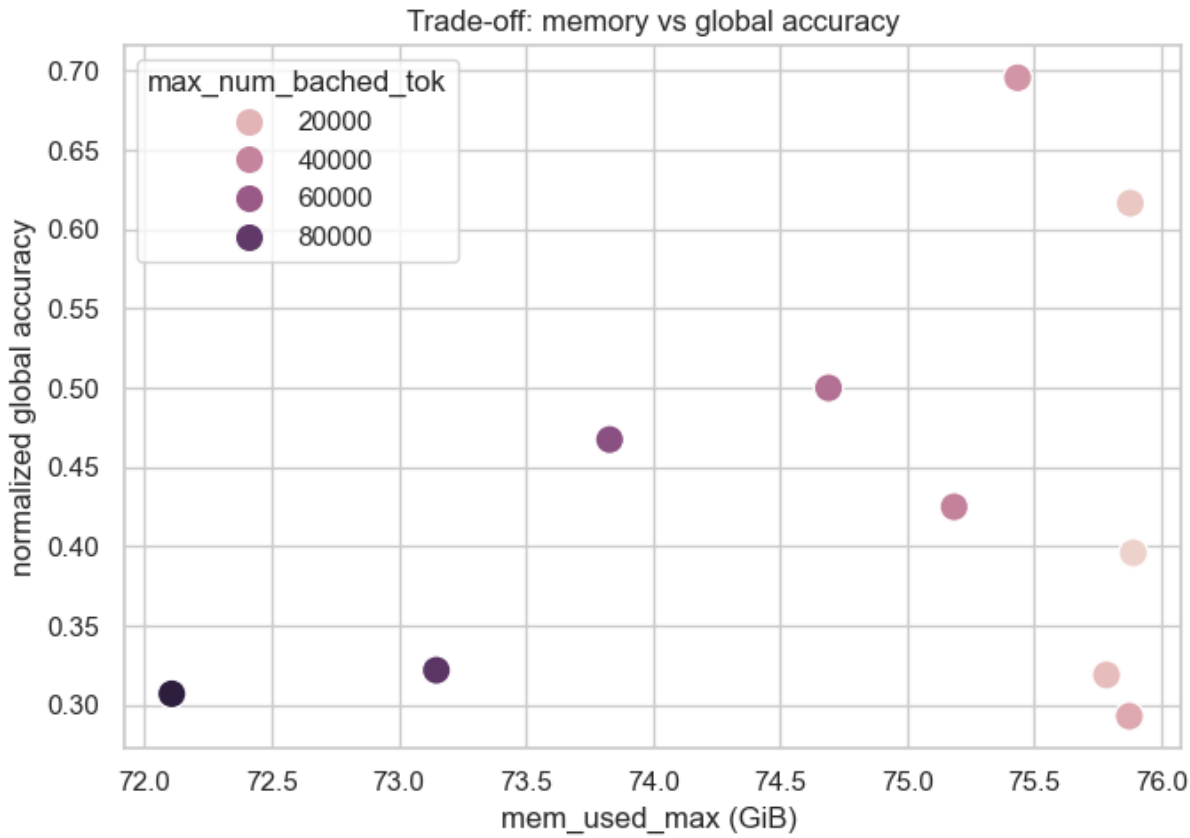
The heatmap confirms that accuracy is very stable across the entire grid:

- ARC is essentially constant.
- HellaSwag and TruthfulQA fluctuate minimally.
- Winogrande and GSM8K show small local variations, but no consistent downward trend at higher values.

Overall, there is no evidence that increasing max_num_batched_tok harms accuracy in this setup.

4) Memory vs global accuracy trade-off

The trade-off plot indicates that we can obtain similar global accuracy values across a wide range of memory levels, and that some of the best accuracy points occur without needing the highest memory configurations. Given the weak sensitivity of accuracy, the practical objective becomes selecting a point that minimizes memory while staying within the "flat" accuracy region.



Conclusion:

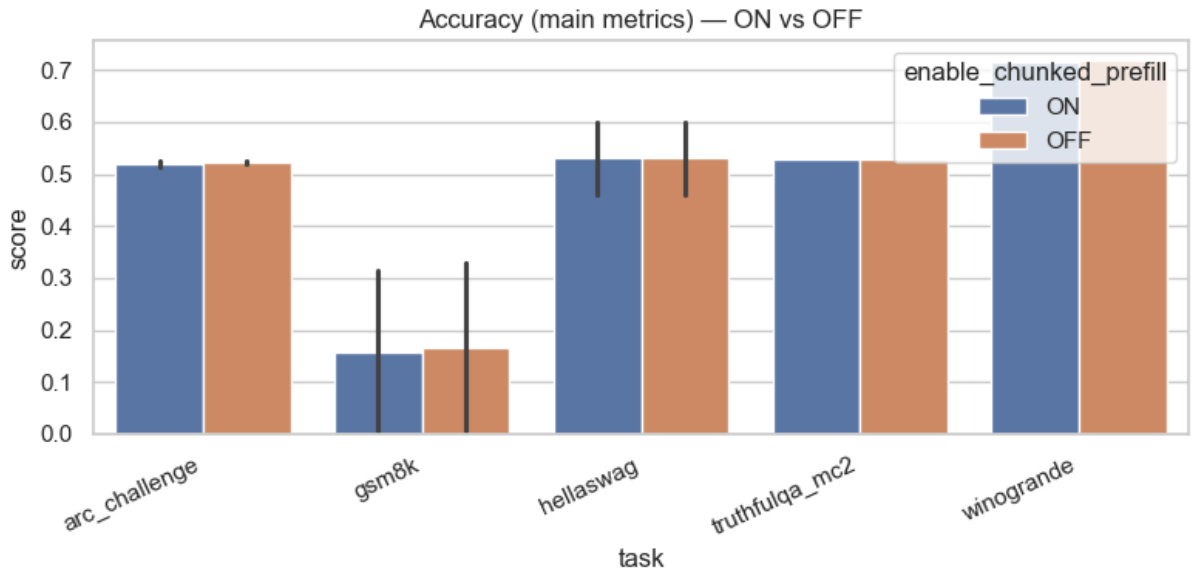
Because accuracy stays nearly unchanged across the sweep and memory improves strongly with larger max_num_batched_tok, the best choice is to move toward the high end of the tested range.

max_num_batched_tok = 65536

- It sits in the region where memory is already significantly reduced compared to low values, providing strong headroom and stability.
- Accuracy remains at the same level as the rest of the sweep (no systematic degradation in the heatmap).
- It is a robust “sweet spot” that avoids relying on an extreme endpoint (98304) unless maximum headroom is strictly required.

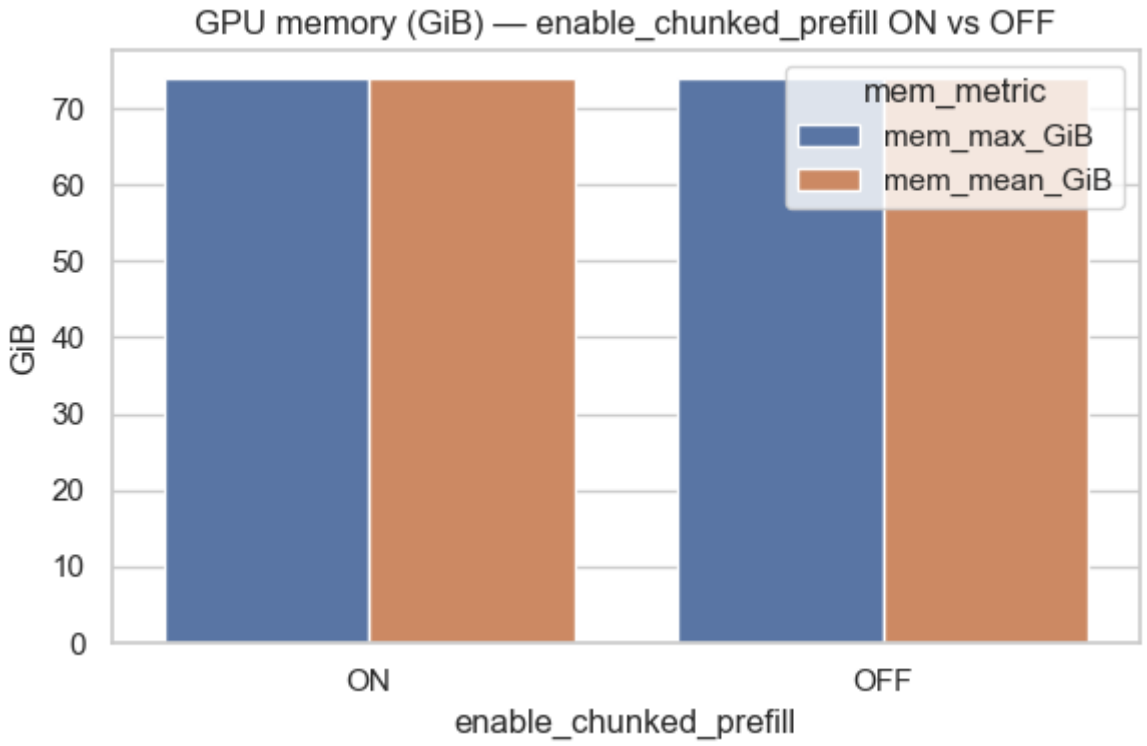
3. Enable_chunked_prefill

Figure 1) Accuracy comparison (ON vs OFF)



To assess whether enable_chunked_prefill impacts model quality, we compare the main accuracy metric per task for the two runs (ON vs OFF). The selected plot is the accuracy heatmap (task vs ON/OFF) because it provides the most direct and readable view of per-task differences. The results are nearly identical across tasks: ARC and HellaSwag are essentially unchanged, TruthfulQA is stable, and GSM8K/Winogrande only show small variations. Overall, the plot indicates no meaningful accuracy difference between ON and OFF under our evaluation setup.

Figure 2) GPU memory comparison (ON vs OFF)



To evaluate memory impact, we use the GPU memory bar plot comparing peak memory (max GPU) and average memory (mean over GPUs) for ON vs OFF. Both mem_max_GiB and mem_mean_GiB are virtually the same between the two configurations, with differences on the order of ~1–2 MiB, which is negligible compared to the overall VRAM usage (~74 GiB). This confirms that enable_chunked_prefill does not materially affect GPU memory consumption in this experiment.

Table 3) Performance metrics (ON vs OFF)

	metric	ON	OFF	OFF_minus_ON	OFF_vs_ON_%
0	total_tok_s	2659.843088	2527.903093	-131.939995	-4.960443
1	completion_tok_s	2603.899926	2474.734959	-129.164966	-4.960443
2	prompt_tok_s	55.943162	53.168134	-2.775029	-4.960443
3	lat_p50_s	3.032010	3.292686	0.260676	8.597462
4	lat_p95_s	3.383452	3.380126	-0.003327	-0.098329
5	lat_p99_s	3.404974	3.402305	-0.002669	-0.078395
6	lat_mean_s	3.113784	3.280260	0.166477	5.346441
7	wall_time_s	15.730251	16.551267	0.821016	5.219345

We include the performance summary table (throughput and latency) to quantify runtime effects. While accuracy and memory are essentially unchanged, the table

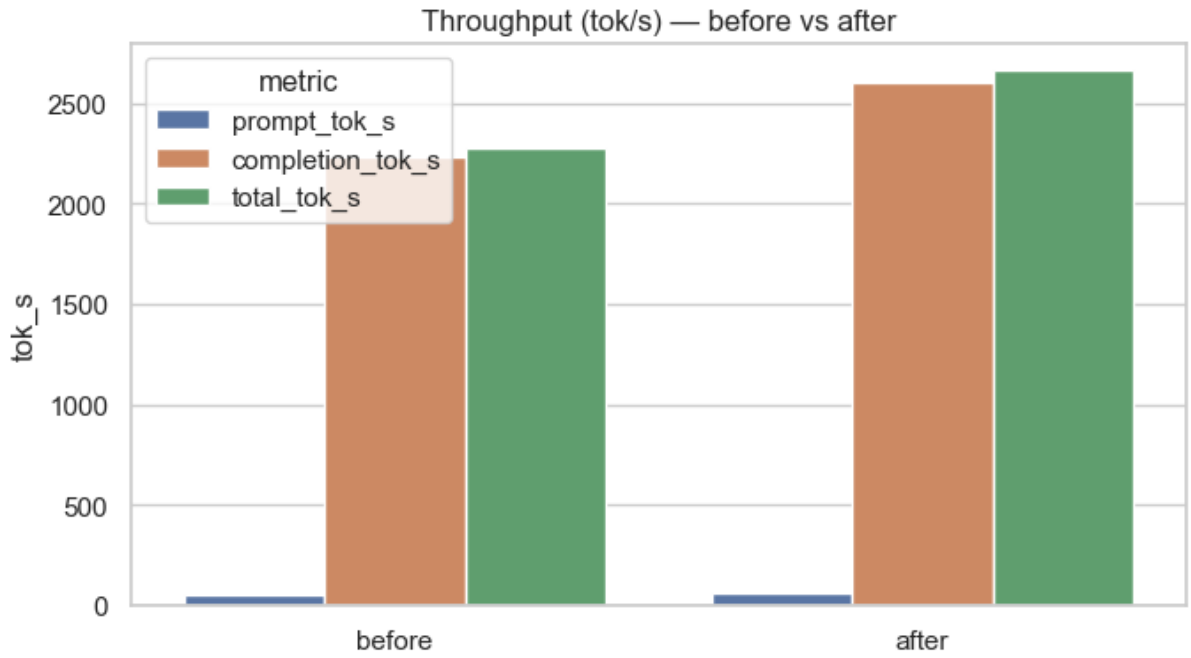
shows that enable_chunked_prefill=ON delivers a ~5% higher throughput (tok/s) and improves typical latency (p50 / mean), with p95/p99 remaining almost identical. This suggests that, although the parameter does not influence memory or accuracy, it can provide a measurable performance benefit.

Conclusion

In this ON vs OFF comparison, enable_chunked_prefill has no significant impact on either accuracy or GPU memory. Any observed differences in per-task scores are small and consistent with normal evaluation variability, and memory differences are negligible. Therefore, from a memory/accuracy perspective, the setting is not a deciding factor. However, the performance table indicates that enabling chunked prefill can improve throughput and median latency, making enable_chunked_prefill=ON the preferred choice when performance is also considered.

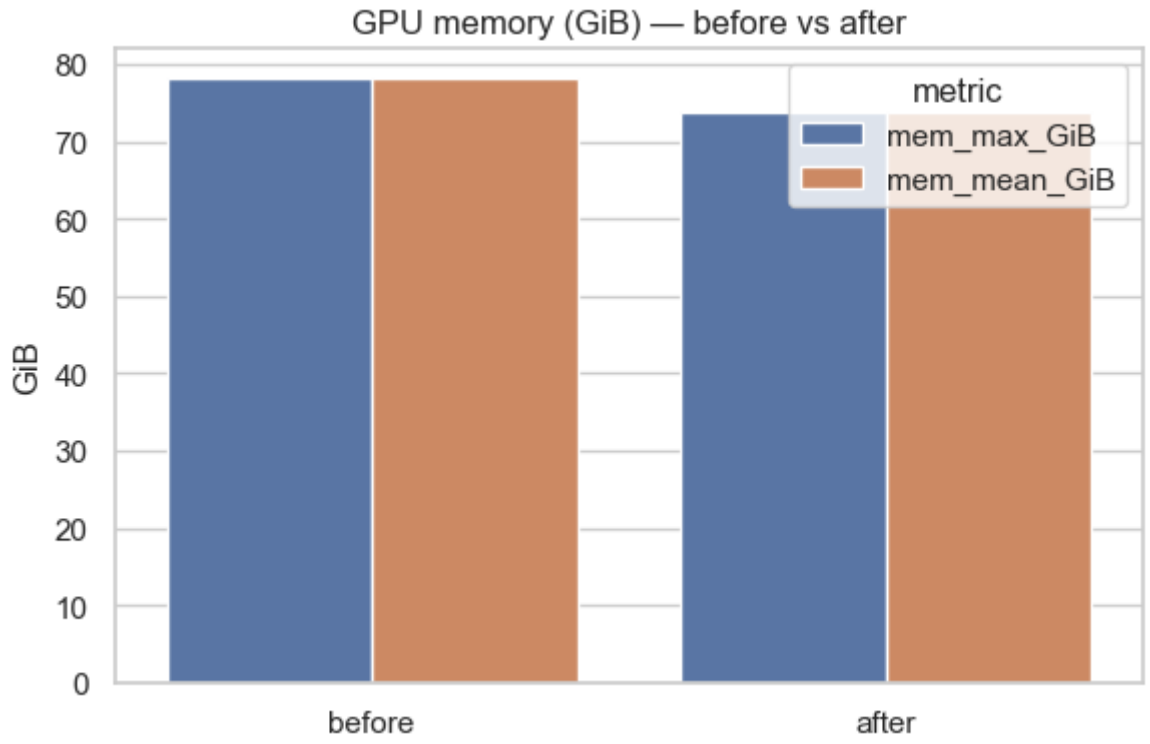
4. Interpretation of the before vs after comparison

Throughput (tok/s)



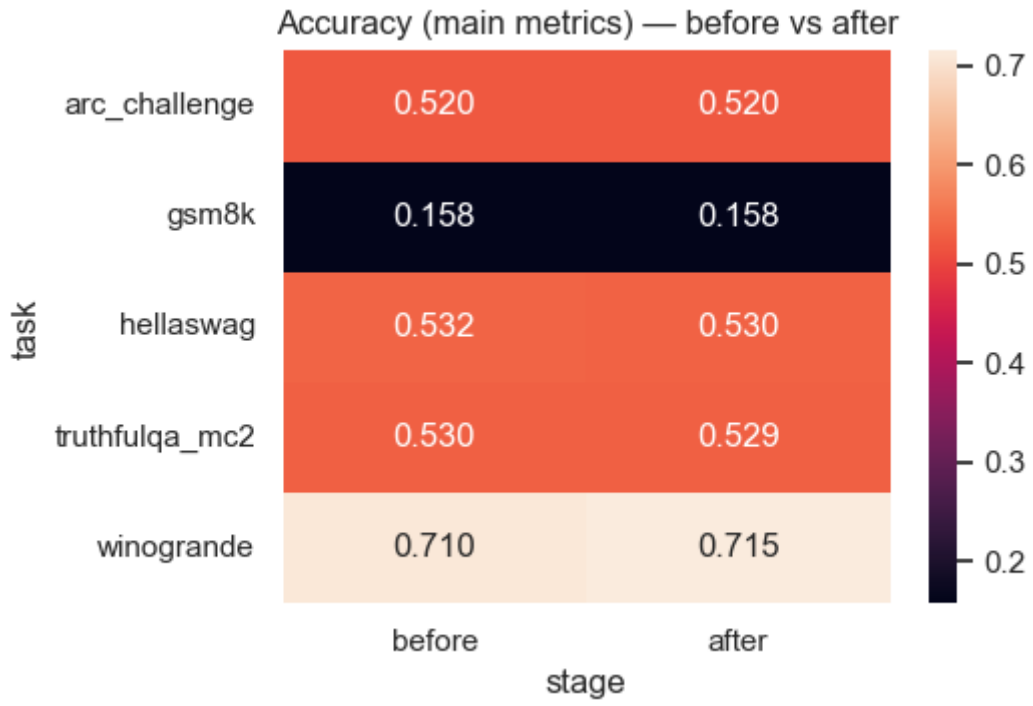
The throughput plot shows a clear improvement after optimization. Both total_tok_s and completion_tok_s increase substantially (about **+16–17%**), and prompt_tok_s follows the same trend. This indicates that the optimization improves overall serving efficiency, not only a single component of the pipeline.

GPU memory (GiB)



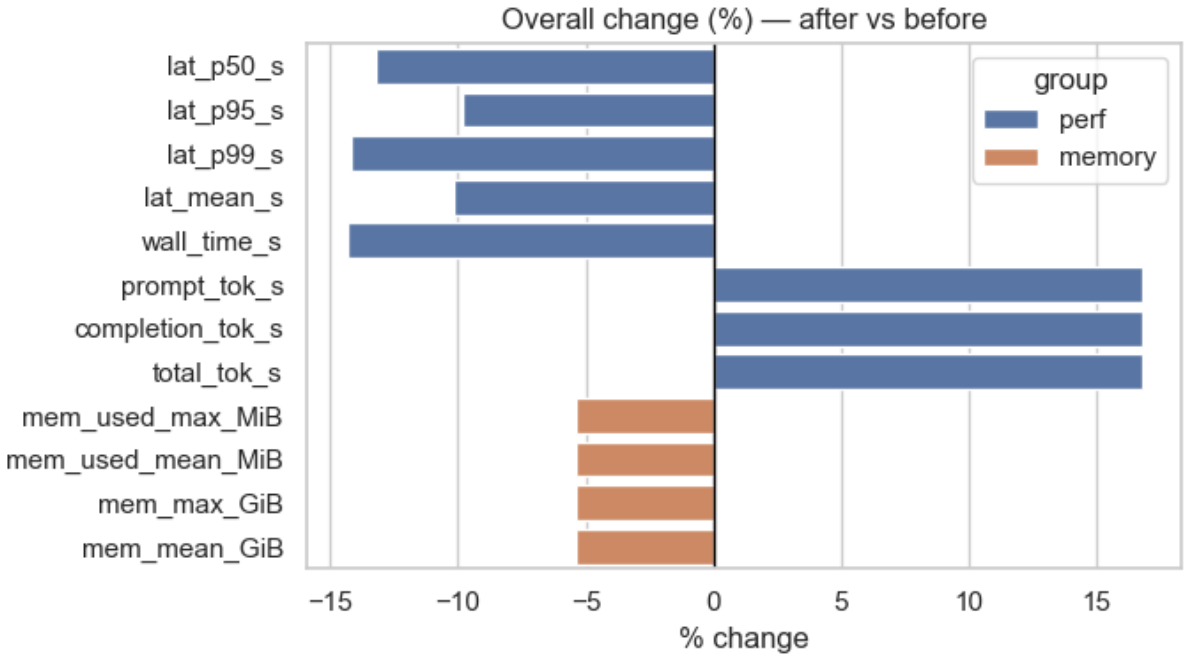
The GPU memory plot shows a meaningful reduction in both peak and average memory usage. mem_max_GiB and mem_mean_GiB decrease by approximately **5.4%**, corresponding to about **4.2 GiB** of VRAM saved. This provides additional memory headroom and reduces the risk of memory pressure or OOM events under heavier load or longer-context scenarios.

Accuracy (main metrics)



The accuracy heatmap confirms that quality remains essentially stable after optimization. ARC and GSM8K are unchanged, while HellaSwag shows only a very small decrease and Winogrande a small increase. Overall, the differences are minor (on the order of a few thousandths), which is consistent with normal evaluation variability for a limited sample size.

Overall change (%)



The summary plot highlights a favorable trade-off: throughput increases while latency metrics decrease across the distribution. In particular, p50, p95, p99, and mean latency are all reduced (roughly ~10–14%), confirming that the optimization benefits not only typical requests but also tail latency.

Conclusion

Overall, the optimized configuration delivers a strong system-level improvement with no meaningful loss in accuracy. Compared to the baseline, it achieves ~+16–17% higher throughput, ~10–14% lower latency (including p95/p99), and ~5.4% lower GPU memory usage (~4.2 GiB saved). Accuracy remains effectively unchanged across the evaluated tasks, with only marginal fluctuations.