

Read Model Configurations and Estimate Memory

Gavin · AI-assisted course design

Contents

1	Goal	3
2	Part 1 — Establish exactly which sources you read	3
3	Part 2 — Translate keys into a computation	4
4	Part 3 — Predict how cache memory changes	5
4.1	A — GPT-2 full-length payload	5
4.2	B — Gemma full-length payload	5
4.3	C — Gemma with bounded local retention	6
5	Part 4 — Estimate weights without loading them	6
6	Part 5 — Optional standard-library calculator	6
6.1	Running the supplied tools	7
7	Checkpoints and analytical answers	8
8	Submit a bounded conclusion	9
9	More Learning	9

1 Goal

Read two small public configuration sources, translate their fields into architecture, and estimate weight and KV-cache payloads. The central skill is separating what a file states, what an implementation establishes, what a calculation assumes, and what an actual run would have to measure.

Prerequisite: Lesson 2.4 — From the Original Transformer to Modern LLMs, especially its head-count and memory sections.

Time: about 60–90 minutes, with another 20–30 minutes for the optional calculator.

Requirements: a browser or previously saved copies of the public text sources, a text editor, and ordinary arithmetic. Optional coding uses an existing Python 3 installation and the standard library on a CPU. No model weights, tokenizer download, machine-learning library, account, API credential, GPU, or paid compute is needed.

Expected artifact: `submission.md` containing your original predictions, evidence table, architecture sketches, memory worksheet, and conclusions. If you implement the optional calculator, also retain its source, input data, and actual output.

! Predict before calculating

Save your predictions before reading the answer section or running a calculator. Keep incorrect predictions and explain their corrections. A numerical answer copied from this Lab is an analytical reference, not evidence of a software run.

2 Part 1 — Establish exactly which sources you read

Use these fixed references:

1. **GPT-2 configuration:** open `config.json` in the configuration repository at revision 7979bb7. The full revision is 7979bb77dc27f20ed20949aa4b91db714c64751f. Read or save only `config.json`, a 665-byte text file. Do not use the page’s model-loading examples or download the repository’s weights.
2. **Gemma 3 1B configuration:** read Google’s `gemma/config.py` at revision cb7c015. The full revision is cb7c0152a369e43908e769eb09e1ce6043afe084. Locate `get_config_for_1b`, then inspect any needed `GemmaConfig` defaults and the `get_model_config` caller. This is a Python configuration factory; read it as text. Do not import or execute downloaded source.
3. **GPT-2 interpretation reference:** use OpenAI’s `src/model.py` at revision c2dae27, full revision c2dae27c1029770cea40978813f17a5fd545b883. Read the `attn`, `block`, and `model` definitions only as needed to interpret the configuration. No TensorFlow installation is needed.
4. **Gemma interpretation reference:** read Google’s `gemma/model.py` at the same cb7c015 revision when a claim requires implementation evidence. `GemmaAttention`, `Gemma2DecoderLayer`, `GemmaModel`, and the cache-allocation portion of `generate` establish the relevant boundaries. Despite its class name, that decoder-layer implementation is also selected for Gemma 3. Read only; no PyTorch installation or execution is required.

The two config sources have different forms. GPT-2's file is serialized model metadata. Gemma's file constructs variant configurations from defaults and overrides. Do not treat an uncalled factory as a loaded model or copy a different variant's fields. The schemas are not interchangeable: the Hugging Face JSON names `vocab_size` and `n_positions`, while the historical OpenAI implementation constructs its token and position tables using `n_vocab` and `n_ctx`. Map these by meaning rather than passing the JSON unchanged to that code. OpenAI's generic `n_vocab=0` is a placeholder; our specimen supplies vocabulary size 50257. The selected JSON's position fields and the chosen position-table length agree at 1024.

If a source cannot be retrieved, record that limitation. You may complete the arithmetic using the lesson's stated values, but label that route as analysis of supplied values rather than direct source verification. Do not silently switch to a similarly named model or an unpinned main revision.

Make an evidence table with these columns:

Claim or quantity	Exact field or source location	Value	Evidence category	Limitation
Example: cache bytes per scalar	Worksheet assumption	2	Assumed	Does not measure a running cache

Use four evidence categories: **stated in source**, **established by implementation**, **derived**, and **assumed**. An observed runtime category is available only if you actually performed a relevant measurement; this core Lab loads no model.

3 Part 2 — Translate keys into a computation

For both specimens, record:

- residual width and block count;
- query-head count, KV-head count, and head width;
- vocabulary size and position-limit field;
- positional mechanism and normalization type;
- local/global attention pattern, if any;
- the source of any dtype value or default you report.

For GPT-2, mark facts that require the implementation rather than a dedicated JSON key. The absence of `num_key_value_heads` does not itself prove a value. For Gemma, distinguish fields set inside `get_config_for_1b` from values supplied by its caller or inherited from the configuration class. Do not run the factory to discover them.

Sketch each model's attention projections with row-vector matrix dimensions. Show the residual width, total query width, total key width, and the width returned by the output projection. Label whether each uses multi-head, grouped-query, or multi-query attention under the definitions in Lesson 2.4.

Answer these questions before checking your sketch:

1. Must residual width equal query-head count times head width?
2. Does sharing a KV pair force the associated query heads to have equal attention probabilities?
3. Which configuration value would determine how many learned positions GPT-2 can index? Why does a RoPE model require a different positional explanation?
4. Does the `get_model_config` caller's default dtype prove the dtype of a model loaded by another program?
5. If the factory declares a local-attention pattern, does that prove that the serving engine evicts old local KV entries?

For the mixed Gemma pattern, number the layers from 1 through 26 and mark each local or global. Keep this list: it will determine the memory calculation.

4 Part 3 — Predict how cache memory changes

Before calculating bytes, predict the effect of each change while keeping the other worksheet assumptions fixed:

- batch size increases from one independent sequence to two;
- stored sequence length doubles under full-length storage;
- KV heads increase from one to four at unchanged head width;
- weights become four-bit while the cache remains BF16;
- local layers retain only their sliding window rather than full-length arrays.

Distinguish a change in the arithmetic model from a valid modification of an existing trained checkpoint. In particular, replacing a KV-head count in a worksheet does not convert a model's learned projections.

Use the following cache assumptions throughout the baseline:

- keys and values both have the specified head width;
- two bytes per stored scalar;
- no quantization metadata, alignment, temporary tensors, or allocator overhead;
- no cross-sequence prefix sharing, device replication, or additional hypotheses such as beams;
- batch one, unless a question changes it;
- all mentioned cache entries have already been processed and are retained.

4.1 A — GPT-2 full-length payload

Use 12 layers, 12 KV heads, width 64, and 1024 stored positions. Write the logical shape of one layer's key array and value array. Compute total bytes, MiB, and GiB from

$$M_{KV} = 2BLS h_{kv} d_h b.$$

Then change only cache precision to four bytes per scalar. Does the parameter count change? Does this calculation establish which cache precision any particular loading command would choose?

4.2 B — Gemma full-length payload

Use 26 layers, one KV head, width 256, and 32,768 stored positions. Calculate full-length payload in bytes and GiB. Repeat with four KV heads as a clearly labeled hypothetical architecture. Explain why substituting four **query** heads in the original calculation would be an error.

4.3 C — Gemma with bounded local retention

Now assume an engine keeps only the latest 512 positions in every local layer, but retains the full prefix in each global layer. Use your layer list to compute

$$M_{KV} = 2Bh_{kv}d_hb [L_gS + L_\ell \min(S, 512)].$$

Evaluate this at $S = 1024$ and $S = 32768$. Compare it with full-length allocation at the same two lengths. Predict which part continues growing once the local layers have reached 512 entries.

These are two different storage assumptions for the same attention pattern. The lesson documents that Google’s pinned reference generator allocates full-length arrays. The bounded calculation estimates a retention policy that another suitable engine could implement; it is not a measurement of that reference program.

5 Part 4 — Estimate weights without loading them

Use the GPT-2 specimen to count unique learned scalars. Its residual width is $d = 768$, vocabulary size is $V = 50257$, position-table length is $T = 1024$, and layer count is $L = 12$. The source has a fourfold MLP expansion, ordinary attention projections with biases, two affine LayerNorms per block, a final affine LayerNorm, and a vocabulary readout tied to the token embedding.

Derive this count instead of treating the marketed size as exact:

$$P = Vd + Td + L(12d^2 + 13d) + 2d.$$

Explain every term:

- token and learned-position embeddings;
- four attention matrices and two MLP matrices per block;
- attention and MLP biases;
- normalization scales and offsets;
- the absence of an additional independently stored vocabulary readout matrix.

Compute weight-only payload for FP32, FP16/BF16, ideal packed eight-bit, and ideal packed four-bit storage. Use integer byte counts before converting to GiB. For the quantized rows, explicitly exclude metadata, mixed-precision exceptions, and padding.

Add the FP16/BF16 weight payload to the two-byte GPT-2 cache payload from Part 3A. Explain why even this combined number remains insufficient as a minimum hardware recommendation.

Finally, consider a separate toy quantization layout with 1024 scalars: four-bit values packed without padding, groups of 64 values, one two-byte scale per group, and no zero points or other metadata. Calculate value bytes, scale bytes, and their total. Which assumption would fail if some tensors stayed FP16?

6 Part 5 — Optional standard-library calculator

Implement a small calculator only after finishing your hand worksheet. The requirements below also describe the supplied standard-library calculator. Save it after completing your worksheet; the published analytical answers remain separate from your actual output.

Create your own compact JSON input files containing the normalized architecture fields needed for the calculations and a source URL plus revision for each specimen. Preserve the original field names in your evidence table. Label your JSON as a learner-created transcription; it is not a publisher's complete config.

The calculator should:

1. Read only these local JSON files; perform no downloads and no model loading.
2. Require explicit `query_heads`, `kv_heads`, and `head_dim`. Do not infer Gemma's head width from division. Require positive integer dimensions and reject non-divisible query/KV grouping for this simplified calculator.
3. Calculate full-length and mixed-retention KV payloads with an explicit per-scalar byte count, layer counts, batch, and sequence length. Require nonnegative local and global layer counts, allow zero local layers for an all-global model, and check that their sum equals the positive total layer count.
4. Calculate ideal weight payload from an explicit unique-parameter count and bits per scalar. Reject non-byte-aligned packed totals or state an explicit ceiling rule rather than silently discarding fractional bytes.
5. Print assumptions, exact bytes, MiB, and GiB separately. Label all results as calculated payloads, not measured RAM or VRAM.
6. Test that batch doubling doubles cache bytes, changing one KV head to four quadruples them, and changing weight bits alone leaves a separately specified cache unchanged.
7. Test a short sequence below the local window: at $S = 128$, bounded-local and full-length payloads should match under otherwise identical dimensions.

6.1 Running the supplied tools

Save `calculate_memory.py` into your Lab directory. The normalized JSON schema uses `residual_width`, `layers`, `query_heads`, `kv_heads`, `head_dim`, `batch`, `sequence_length`, `cache_bytes_per_scalar`, `local_layers`, and `global_layers`. Add `local_window` when local layers exist. For weight estimates, supply `unique_parameters` and `weight_bits_per_scalar` together. Include source with `url` and the complete revision. Each numerical field must be an integer under the validation rules above. These fields are your transcription and assumptions; they are not a publisher config.

```
python3 calculate_memory.py my-normalized-config.json > calculated-payloads.json
```

The program was tested on CPython 3.13.13 and uses no dependencies or network. JSON output separates exact bytes, MiB, GiB, logical projection shapes and exclusions. A zero local-layer count is valid for GPT-2; residual width need not equal total query width.

For optional retrieval, save `read_config_sources.py`. It retrieves exactly five pinned text files: the two configurations, the two interpretation references above, and Lab 09's gpt-oss config. The total is **48,054 bytes**. It checks each fixed size and SHA-256, retains original bytes and embedded notices, and writes provenance. It never imports the Python reference files or retrieves weights/tokenizer files. Read downloaded `.py` files in your text editor.

```
python3 read_config_sources.py --plan
python3 read_config_sources.py --output config-references
python3 read_config_sources.py --offline --output config-references
```

Cached references are verified rather than silently replaced; altered or unexpected files fail. Initial retrieval requires internet; the calculator and verified cached references work offline. Consult each publisher's full license in its pinned repository.

Retain your command, Python version, actual output, and any corrections. A match confirms your calculator against these cases; it does not benchmark an inference engine. Never execute the downloaded model implementation as a shortcut.

7 Checkpoints and analytical answers

i Compare after completing your worksheet

Architecture. GPT-2 has residual width 768, 12 query and KV heads, and head width 64. Its W_Q, W_K, W_V, W_O each have logical shape 768×768 . Gemma 3 1B has residual width 1152, four query heads, one KV head, and explicit head width 256. Its W_Q is 1152×1024 , W_K, W_V are 1152×256 , and W_O is 1024×1152 . Library storage may transpose these logical matrices or fuse Q, K, and V into one projection. Count the same learned scalars only once.

Positions and layers. GPT-2 uses a learned position table. The selected JSON reports `n_positions: 1024`; the paired historical OpenAI implementation uses `n_ctx` to size that table. These names identify the same 1024-position assumption here, not a shared input-file schema. Gemma's factory specifies 32,768 positions and a repeating local/local/local/local/local/global pattern. With one-based numbering, global layers are 6, 12, 18, and 24. The other 22 are local. Query heads sharing KV can still produce different distributions.

Cache A. Each GPT-2 key or value array has logical shape (1, 12, 1024, 64). Across 12 layers at two bytes per scalar, the total is 37,748,736 bytes, 36 MiB, or 0.03515625 GiB. Four-byte cache scalars double this to 75,497,472 bytes. Neither calculation changes learned parameter count.

Cache B. Gemma full-length storage is 872,415,232 bytes, or 0.8125 GiB. The hypothetical four-KV-head version gives 3,489,660,928 bytes, or 3.25 GiB. These are analytical payloads, not observations.

Cache C. At $S = 1024$, bounded local retention gives 15,728,640 bytes, or 15 MiB; full-length storage gives 27,262,976 bytes, or 26 MiB. At $S = 32768$, bounded retention gives 145,752,064 bytes, or 139 MiB, versus 832 MiB for full-length storage. Beyond the local window, only the four global layers' retained payload keeps growing under the bounded policy.

Weights. Attention contributes $4d^2$ matrix scalars per block, and a width- $4d$ two-matrix MLP contributes $8d^2$. Their biases contribute $4d + 5d$, and two LayerNorms contribute $4d$, totaling $13d$. The embedding, positional, and final-normalization terms complete $P = 124,439,808$ unique learned scalars.

Weight assumption	Exact payload bytes	Approximate GiB
FP32	497,759,232	0.463574
FP16 or BF16	248,879,616	0.231787
Ideal packed eight-bit	124,439,808	0.115894
Ideal packed four-bit	62,219,904	0.057947

Two-byte weights plus the two-byte GPT-2 cache total 286,628,352 bytes, approximately 0.266943 GiB. This excludes activations, workspace, runtime overhead, and any additional tensors or copies. It is not measured process memory.

Toy quantization overhead. Packed values occupy 512 bytes. Sixteen two-byte scales add 32 bytes, giving 544 bytes total. Uniform four-bit storage fails if some tensors remain FP16; count those separately. Quantizing weights does not change an independently specified cache dtype.

Metadata limits. A dtype field, a cache flag, and a local-attention pattern do not establish the loaded tensor dtype, actual cache allocation, or eviction policy. Those require matching implementation evidence and, for an observed-state claim, a real measurement.

8 Submit a bounded conclusion

Finish with three short paragraphs:

1. Explain one architectural difference whose memory consequence you calculated.
2. Identify one tempting inference that the configuration alone does not justify.
3. State which additional evidence you would need to estimate peak inference memory or compare speed on a specific machine.

Include the original predictions, exact revisions, evidence categories, dimensions, assumptions, and units. If your optional calculator disagrees with the hand calculation, show the first divergent quantity and the correction. Do not invent loaded-model outputs for a Lab that never loads one.

9 More Learning

1. **GPT-2 configuration snapshot** Compare the small configuration file with the much larger checkpoint artifacts without downloading the latter.
2. **Google Gemma configuration factory** Practice resolving one variant's overrides and inherited defaults.
3. **OpenAI GPT-2 implementation** Verify the tied vocabulary readout and the parameter-count terms from their tensor shapes.
4. **Transformers cache strategies, v4.51.3** Compare growing, static, offloaded, and quantized cache policies. Separate memory placement and allocation from the model's attention mask.