

6.2 — The LLM Microscope

Gavin · AI-assisted course design

Contents

1 Turn a successful experiment into something you can inspect again	3
2 Begin with a question and a specimen	3
3 Keep the architecture small enough to understand	4
4 Give every measurement an address	6
5 Observation and intervention need different interfaces	6
6 Preserve exact identity without collecting everything	7
7 Store results so analysis can be repeated independently	7
8 Treat incomplete execution as a first-class outcome	8
9 Make analysis match the question	8
10 Numerical agreement is a declared test	9
11 Keep the core local and the output private by default	9
12 Write the conclusion at the scale of the evidence	10
13 Questions to explain back	10
14 More Learning	11
15 Ask your own question	11

1 Turn a successful experiment into something you can inspect again

You return to an activation experiment a month later. A plot shows that an intervention changed a token score, but its filename says only `results-final`. Which checkpoint produced it? Was the vector captured before or after normalization? Did the script change every token or only the last one? Were the displayed examples chosen before the experiment? The plot cannot answer these questions.

An LLM microscope is a small, reusable arrangement of tools that keeps those answers attached to the evidence. It joins a model, a precisely located measurement, an optional intervention, a bounded execution plan, and an inspectable result. Its most important feature is that you can follow a conclusion back to the computation that supports it.

The course has already supplied its pieces. Tokenization identifies the model's actual input. Architecture determines valid measurement boundaries. Inference engineering controls execution. Interpretability distinguishes observation from intervention. Experimental design determines which comparison can answer a question. Now we connect these pieces without building a large framework.

Lab 6.2 wraps the small CPU experiments from Labs 5.1 and 5.3 in one command-line workflow. It reuses their saved measurements and adds one fixed integration check. This lesson and Lab specify a tool to implement and verify; they do not assert that a finished microscope or successful measurement already exists.

By the end, you should be able to design a minimal run record, separate capture from intervention, validate reuse of earlier results, regenerate an analysis without rerunning a model, and decide whether a claim is supported, unresolved, or blocked by failed checks.

i Math to know / refresh

Needed now: vector norms, paired differences, means, and elementary memory accounting. We revisit them where the pipeline needs them.

Useful refresh: logits, cosine similarity, variance, and uncertainty from Lesson 6.1. The microscope does not introduce a new universal metric.

Side trail: larger-model instrumentation, sparse routing, and experiment registries. These are extensions after one small workflow is trustworthy.

2 Begin with a question and a specimen

A microscope needs a specimen. "A Transformer" is insufficient because architecture, weights, tokenizer, software, and execution settings jointly determine the observed computation. Replacing any one of them may change a result while leaving the displayed model name unchanged.

The core specimen remains EleutherAI/pythia-14m, at revision 94f7c35d5e9f2e9bac8ca839329f505b4d007d5d. Its configuration describes a dense six-block model of width 128 with parallel residual updates. It is useful here because the course has already developed

bounded observation procedures for it. Its usefulness as an instructional specimen does not depend on impressive answers. Immutable publisher revision

Write the question separately from the specimen. For example: “Can the integrated runner reproduce its unchanged baseline after installing and removing observation and intervention hooks?” This is a valuable engineering question. “Does the frozen direction increase the specified logit contrast on these prompts?” is a different experimental question. An affirmative answer to the first does not predetermine the second.

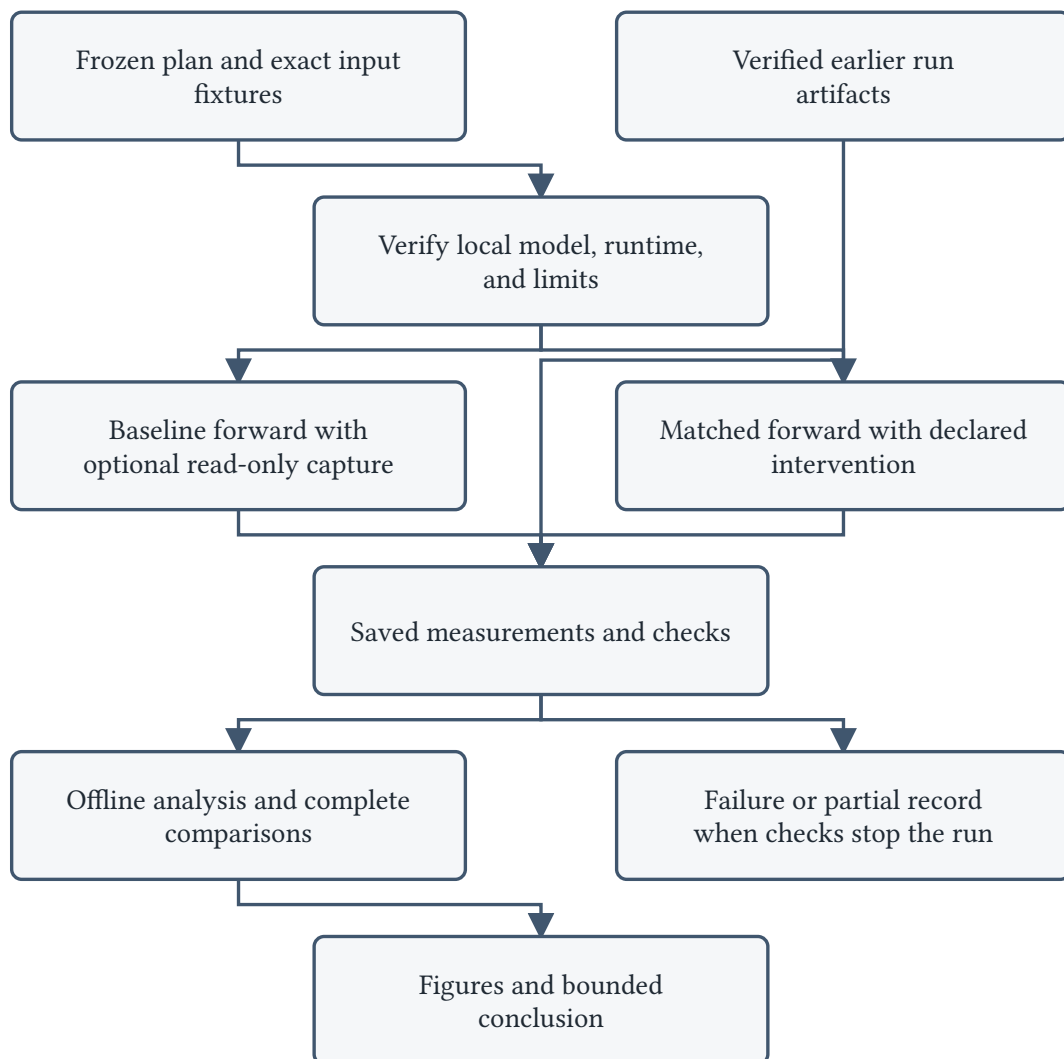
Start each run with a configuration containing exact prompt fixtures, selected boundaries, interventions, metrics, tolerances, limits, and prediction records. Freeze those choices before the relevant observations. The previous lesson explained why exploratory choices and confirmatory tests need different treatment. A reusable program must preserve that distinction rather than hide it behind an attractive report.

3 Keep the architecture small enough to understand

A useful first implementation has four pieces. A **model adapter** loads the verified local specimen and describes its valid boundaries. A **capture routine** copies selected values without changing them. An **intervention routine** makes one explicitly described replacement. An **analysis routine** reads saved data and produces summaries. A short command-line coordinator validates the configuration and calls these pieces.

The coordinator should know how to count attempts, create a run directory, and stop. It should not invent prompts, select a promising layer, install dependencies, or provision a larger machine when something fails. Scientific choices remain visible in the plan. File handling remains visible in the implementation.

Here is the complete information flow. The diagram is an original course design; the two forward paths emphasize that an intervention requires another execution from the same specified input.



Read the diagram as a provenance chain: the frozen plan identifies what will happen; two comparable executions supply evidence; saved records feed analysis. Imported records retain their original identities. A failure branches into an honest record rather than disappearing from the picture.

Avoid an abstract plugin system that claims to support every model. One explicit adapter and a few ordinary functions are easier to test. Add another adapter only when a concrete experiment needs it. Keeping code small also makes it easier for a learner to explain what each hook actually does.

💡 Lab pause — Define the smallest useful instrument

Complete Lab 6.2, Parts 1–2. Inventory the evidence you already have and write the proposed command contract. Identify which commands may perform model forwards and which must work entirely from files.

4 Give every measurement an address

A tensor shape identifies its dimensions, not its meaning. Several values in our specimen have width 128: embeddings, raw block outputs, normalized states, and branch contributions. A vector saved as `layer3.npy` leaves too much unspecified.

A measurement address should include the model revision, implementation version, module path, input or output boundary, normalization status, zero-based block index, batch row, token position, and token ID. It also needs source and stored shapes, dtype, and capture mode. This permits an auditor to distinguish the source tensor $(1, n, 128)$ from a saved final-position vector $(128,)$.

In the pinned implementation, the raw output of `gpt_neox.layers[2]` is the first element of the block's returned tuple. This is the third block, after its residual addition. Final normalization happens later. Verify the installed source and runtime object before using that address. GPT-NeoX implementation, Transformers 4.57.1

Raw-state names from Lab 5.1 remain useful: r_0 enters block 0, $r_{\ell+1}$ leaves block ℓ , and h_f names the separately normalized final state. Reuse those names instead of creating a new numbering convention during integration. Earlier ambiguities become harder to fix once many plots depend on them.

The same discipline extends to sparse routing. A future MoE adapter would identify router scores, selected expert IDs, associated weights, token indices, and whether weights precede or follow selection and renormalization. It would document capacity handling and any dropped assignments. Our dense Pythia model has no MoE router. A routing field therefore says `not_applicable`, with a reason; it never contains invented expert IDs or a fabricated empty “measurement.”

5 Observation and intervention need different interfaces

An observation routine copies a value and leaves the forward computation untouched. An intervention routine intentionally returns a changed value for subsequent computation. Using distinct functions makes that difference easy to review and test.

For observation, validate the returned structure, select the requested position, detach and clone it, and return no replacement. For intervention, clone the relevant output tensor, change only the permitted row, and reconstruct the module's expected output structure. Preserve the original values for comparison. These requirements belong to the instrument's implementation contract; Python does not infer them from a function called `observe`.

PyTorch forward hooks can return a replacement output and provide handles for removing registered hooks. The framework mechanism therefore permits both measurement and modification. It does not guarantee that a particular callback is read-only. PyTorch hook reference

Test read-only behavior by comparing uninstrumented logits with instrumented logits under a declared numerical tolerance. Test zero intervention through the actual replacement path. Then remove hooks and repeat an unchanged baseline. Finally, deliberately raise an exception inside a temporary hook and verify cleanup before another baseline. The exception is expected test behavior, not a model result.

Remove handles in a `finally` block. A process killed by a watchdog may not execute Python cleanup; record that uncertainty rather than claiming verified restoration. Ending the worker process discards

its in-memory model. Never reuse that worker after uncertain cleanup. This is why both an execution log and post-cleanup checks matter.

6 Preserve exact identity without collecting everything

Separate three manifests. The **artifact manifest** identifies model and tokenizer files by names, byte counts, and cryptographic hashes. The **source manifest** identifies the runner, configuration, and imported adapter code. The **environment record** captures actual package versions, operating system, processor, device, dtypes, effective threads, and execution settings.

A model label or a mutable branch name cannot replace an immutable revision. A revision cannot replace verification of the local files actually loaded. Likewise, a requirements file describes intended dependencies; the environment record describes the installation that executed the experiment.

A SHA-256 digest is a compact identifier for file contents. Hash the exact bytes and compare against the expected record before consumption. A changed byte changes the digest with overwhelming practical likelihood. Python provides SHA-256 through `hashlib`. [Python hashing reference](#)

Hashes establish a relationship between bytes and a recorded digest. They do not prove who authored the file, whether it was measured honestly, or whether an experiment was well designed. A manifest created today for an old untracked file must say that its integrity history begins today. It cannot retroactively prove that predictions were frozen before evaluation.

Save only what the question needs. One float32 vector of width 128 occupies $128 \times 4 = 512$ raw bytes. Lab 5.1's twenty selected vectors require 10,240 bytes per prompt before metadata. Saving every position, vocabulary score, and layer multiplies storage rapidly. The smallest useful record is easier to inspect, transfer, and keep private.

7 Store results so analysis can be repeated independently

A run directory should be understandable without executing its Python files. Use plain JSON for metadata, JSON Lines for chronological events, and numeric arrays in a transparent format. A numeric NPZ archive can be inspected with pickle disabled; object arrays should be rejected. NumPy warns that loading pickled objects can execute code. [NumPy loading reference](#)

Store original float32 measurements separately from float64 analysis values. A later summary may improve numerical handling without pretending that the model itself ran in float64. Record the analysis source hash so two summaries can be compared when the calculation changes.

Every record needs provenance. A synthetic prompt is an authored input fixture. Its captured activation can still be a real measured value. A hand-designed activation used to test a plot is a numerical fixture, not a measurement. A recomputed statistic is derived evidence whose parents should be named. These labels concern how each object was obtained, not whether its contents look plausible.

Imported measurements retain their original run IDs, statuses, prompt IDs, and file hashes. Importing does not make them fresh samples. Two plots derived from the same activation array remain two views of the same evidence. Replaying a known prompt validates execution consistency; it does not restore held-out status after the earlier result has been inspected.

Keep plots downstream of the records. If a figure is lost, analysis can regenerate it without another model call. If a conclusion changes, preserve the previous report and identify the revised analysis. Do not overwrite raw measurements to make a revised interpretation easier to tell.

8 Treat incomplete execution as a first-class outcome

A run moves from planned to running and then to a terminal state. **Completed** means its required attempts and checks finished according to the plan. **Failed** means an unexpected error or invalid check blocked that plan. **Partial** means a limit, cancellation, or interruption left required work unfinished. A planned run that never executes stays `not_run`.

These are execution states. Scientific outcomes need a separate field: a hypothesis can be supported on the defined metric, unsupported, or unresolved. A completed experiment can produce an effect opposite to the prediction. A partial experiment cannot manufacture the missing comparisons by treating absent values as zero.

Write an attempt-start event before calling the model, then a finish event with the actual outcome. Count attempts, including a call that raises before returning logits. A start with no finish identifies an interrupted attempt. Preserve completed rows and explicitly mark missing ones so a report can show its denominator.

This is also where limits become enforceable. A plan fixes maximum input length, attempts, elapsed execution time, and output bytes. Reject excessive input rather than silently truncating it. Stop rather than widening a layer sweep. Use a watchdog for a stalled pass as well as checks between passes; otherwise a nominal time limit may never be consulted.

These controls are ordinary program safeguards, not a security sandbox. A Python process with filesystem access still has that access. Apply the governance habits from Lab 4.4: validate supported requests, reject unknown operations, and keep external actions outside the measurement loop.

 Lab pause — Break the bookkeeping on purpose

Complete Lab 6.2, Parts 3–4. Test corrupted imports, missing records, fixture-only data, and attempts beyond the cap without executing the model. Confirm that failed checks survive in the output instead of becoming empty success reports.

9 Make analysis match the question

For a paired intervention, let $y_j^{(0)}$ be the baseline outcome and $y_j^{(a)}$ the outcome under arm a on the same prompt. Calculate

$$\Delta y_j^{(a)} = y_j^{(a)} - y_j^{(0)}, \quad \overline{\Delta y}^{(a)} = \frac{1}{N} \sum_{j=1}^N \Delta y_j^{(a)}.$$

Subtract within prompt before summarizing. This preserves the relationship between each intervention and its appropriate baseline. Comparing an intervention on one prompt with a baseline on another answers a different question.

Lab 5.3 defines $y = z_{\text{good}} - z_{\text{bad}}$. Preserve its tokenizer-verified token IDs, direction orientation, dose, and four-review primary endpoint when importing that experiment. Do not silently substitute generation sentiment, average only favorable rows, or combine the off-task prompts into that endpoint.

A transparent plot can show every paired change, the mean, and a zero reference line. Identify prompt families and display negative, near-zero, and missing outcomes. A second plot can show perturbation magnitude or distributional change. Label axes with their actual quantity: a logit difference is neither accuracy nor a probability.

Aggregation needs a sampling argument. Several variants from one template are not automatically independent evidence about many domains. Repeating deterministic analysis of the same files adds no observations. A two-prompt integration replay does not inherit the inferential strength of a larger experiment because it shares its directory format.

For geometrical summaries, retain boundary names and separate raw residuals from final normalization. If cosine similarity has a zero denominator, store an explicit undefined value. Displaying zero would turn an unavailable comparison into a specific geometric claim. Analysis code should refuse non-finite values in quantities that require finite measurements.

10 Numerical agreement is a declared test

A practical comparison uses both absolute and relative tolerance:

$$|a_i - b_i| \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}}|b_i|.$$

The absolute term controls discrepancies near zero; the relative term scales with the reference magnitude. State which vector is the reference and save maximum absolute error alongside pass or fail. A rounded table can conceal an error, so checks must use unrounded values.

Within one fixed CPU environment, the Lab applies its prespecified criteria to observer invariance, zero edits, restoration, and source-compatible replay. Across environments, mathematical equivalence does not guarantee bitwise identity. PyTorch documents numerical differences across platforms, releases, and computation arrangements, and its reproducibility guidance qualifies what seeded execution can guarantee. Numerical accuracy; Reproducibility

When a comparison fails, preserve the failed record. Check checkpoint identity, tokenization, module boundary, dtype, flags, and hook cleanup before changing a tolerance. If a justified change is made, record a new analysis or plan. Quietly relaxing the threshold after seeing the discrepancy destroys the meaning of the original test.

11 Keep the core local and the output private by default

The core requires no new weights, hosted inference, GPU, or AWS resource. Reuse the verified local cache and installed environment. Set offline controls before loading and require local-only loading; the experiment should fail clearly if its specimen is unavailable. Hugging Face documents these offline-loading mechanisms. Transformers offline setup

A larger future question may genuinely require different compute. First justify the needed memory or capability, then obtain approval for the provider, resources, region, spending limit, and lifecycle plan.

Save required evidence before shutdown, verify the instance state, and separately inspect retained storage and other billable resources. Stopping EC2 does not remove all associated storage costs. Termination can permanently delete configured volumes. EC2 stopping; EC2 termination

Budget alerts help notice spending; they are not hard spending caps. AWS documents billing and notification delays, during which costs can exceed a threshold. A watchdog, explicit resource cleanup, and budget alerts address different failure modes. None justifies automatically provisioning compute for this Lab. AWS Budgets

Local results also need a publication boundary. Logs can expose prompts, home-directory paths, usernames, or unrelated configuration. Use synthetic course prompts, collect only necessary runtime fields, and never dump all environment variables. An export operation should copy an explicit list of reviewed files into a separate folder. Creating that folder does not upload or publish it. Review licenses and audience before any later sharing.

12 Write the conclusion at the scale of the evidence

A good report names the question, specimen, plan, checks, observations, and remaining alternatives. Its causal statement describes the exact intervention and outcome. “Adding this frozen vector at this boundary changed this score on these inputs” is stronger than a visual association and narrower than discovering a universal semantic mechanism.

A microscope improves the traceability of such a statement. It cannot supply missing controls, turn a fixture into a measurement, establish subjective experience, or make an unrepresentative sample representative. Successful software integration and successful scientific prediction deserve separate sentences.

Lab pause — Finish the evidence chain

Complete Lab 6.2, Parts 5–7. Run the bounded demonstration only after its prerequisites pass. Regenerate the report from saved data, explain one limitation that better tooling cannot remove, and review the explicit export list.

13 Questions to explain back

1. Why are a model revision and a runtime manifest both necessary?
2. What evidence distinguishes observation-only capture from an unnoticed intervention?
3. Why can an imported activation be measured evidence without being a new sample?
4. Can a completed run contain a failed scientific prediction?
5. What is wrong with converting a missing effect measurement into zero?
6. Which extra evidence would be needed before calling a steering direction a general semantic mechanism?

Suggested answers

1. The revision identifies a published specimen; runtime settings and implementation determine how its local verified files were executed.
2. Inspect callback behavior, compare unhooked and observed logits, test cleanup and restoration, and preserve numerical discrepancies.
3. Its original execution may be genuine, but copying or reanalyzing it does not create an independent observation.
4. Yes. Completion concerns execution and checks; the predicted effect may be absent or reversed.
5. Zero asserts a measured absence of change. Missing means the comparison is unavailable and changes the eligible denominator.
6. Appropriate controls, broader held-out evaluation, specificity tests, alternative explanations, and evidence connecting the edit to ordinary model computation. A favorable score alone is insufficient.

14 More Learning

- PyTorch module hooks: inspect the exact callback and removal interfaces used by your adapter.
- Versioned GPT-NeoX source: trace one chosen measurement boundary through the implementation.
- PyTorch reproducibility guidance: compare its qualified guarantees with what your report actually promises.
- NumPy array loading: understand why transparent numeric arrays and disabled pickle belong in the artifact contract.
- Towards Best Practices of Activation Patching: revisit how measurement and intervention choices shape a causal interpretation.

15 Ask your own question

Curriculum v6 closes with a learner Research Capstone: **Ask Your Own Question**. The optional closing part of Lab 6.2 provides a small starting point. Choose one instructional question, state what evidence could change your mind, reuse suitable tools, and document what you learn. A careful replication, bounded comparison, or informative negative result is enough.

The next step can be modest: explain one unexpected plot, test one alternative wording family, or compare two explicitly defined measurement choices. Freeze a new plan before new observations and keep exploratory work labeled. This is an invitation to practice inquiry, not a requirement to claim a novel discovery. The course has given you a way to ask a precise question and leave an honest record of the answer.