

Test Causal Interventions

Gavin · AI-assisted course design

Contents

1 Goal	3
2 Part 1 — Account for a toy intervention	3
3 Part 2 — Lock the specimen and environment	3
4 Part 3 — Save exact fixtures before observing results	4
5 Part 4 — Identify and validate the exact boundary	5
5.1 Required checks, using discovery inputs only	6
6 Part 5 — Construct the direction using discovery only	6
6.1 Freeze predictions before the held-out pass	6
7 Part 6 — Run all held-out arms	7
8 Part 7 — Measure the intended effect and collateral change	8
9 Part 8 — Implement a reproducible runner	8
10 Explain the result	9
11 More Learning	10
12 Optional implementation files	10

1 Goal

Construct a contrastive residual direction, intervene in an actual trained model, and measure what changes. The goal is a controlled result, including a negative or inconclusive result. A tiny language model is not guaranteed to exhibit clean semantic steering.

Prerequisites: Lesson 5.3, the observation checks in Lesson 5.1, and the bounded model setup from Lab 05.

Time: 90–150 minutes once dependencies and the small checkpoint are available.

Requirements: CPU, the existing local six-file Pythia-14M artifact, and Python with PyTorch and Transformers. No new large checkpoint, paid service, inference API, GPU, account, or training is required. This Lab specifies a runner to implement and execute; the specification is not an already-executed notebook or supplied model result.

Expected artifact: a reproducible runner, frozen predictions, a complete machine-readable run record, and a short interpretation. The analytical warm-up cannot substitute for the trained-model intervention.

2 Part 1 — Account for a toy intervention

Use the hand-designed vectors

$$x = [2, 1]^T, \quad u = [1, 0]^T, \quad s(x) = x_1 + 2x_2.$$

Before calculating, predict the readout after adding u , subtracting u , and removing the projection onto u . Calculate the perturbation norm for every case. Then repeat the comparison for $x = [-2, 1]^T$. Explain why projection removal does not always equal subtracting a fixed positive vector.

These values are an exact construction, not trained-model measurements. Keep its result in a separate section of your submission.

3 Part 2 — Lock the specimen and environment

Use EleutherAI/pythia-14m at revision 94f7c35d5e9f2e9bac8ca839329f505b4d007d5d, reusing the cache prepared for Labs 05/07. The six approved artifacts are config.json, tokenizer.json, tokenizer_config.json, special_tokens_map.json, generation_config.json, and model.safetensors. Do not redownload if the verified cache exists. The safetensors file is 28,143,920 bytes; the six-file set is approximately 30 MB. A missing cache is a setup blocker to report, not permission to fetch another model.

Record the local artifact manifest and SHA-256 of all six files, checking against the earlier verified manifest where available. Read the local configuration and assert six layers, hidden size 128, and use_parallel_residual=True. The immutable publisher revision identifies the specimen.

Use the existing environment: Python 3.13.13, PyTorch 2.9.0, Transformers 4.57.1, tokenizers 0.22.1, safetensors 0.6.2, and huggingface-hub 0.35.3. Record full installed version strings, including build suffixes. If a version differs, stop the confirmatory run and explain the mismatch before treating another environment as equivalent.

The runner must:

- Load only the local directory, with `local_files_only=True`, `trust_remote_code=False`, and safetensors enabled; use the built-in `GPTNeoXForCausalLM` and fast tokenizer.
- Run on CPU in float32, `model.eval()`, `torch.inference_mode()`, and eager attention. Set `use_cache=False`, `output_attentions=False`, `output_hidden_states=False`, and `logits_to_keep=1` explicitly for every pass. Assert returned logits have shape `(1, 1, model.config.vocab_size)` and save only the final-position vocabulary vector.
- Fix PyTorch CPU threads to one, set the global seed to 19, enable deterministic algorithms, and record operating system and processor details. Do not claim bitwise reproducibility across different platforms.
- Set `HF_HUB_OFFLINE=1` and `TRANSFORMERS_OFFLINE=1` before loading. No network access is needed during this experiment.
- Count every attempted forward call, including the deliberately aborted cleanup check. Refuse a ninety-first call. Enforce a fifteen-minute cumulative compute wall-clock limit across discovery and evaluation, excluding prior setup and time spent writing predictions; carry elapsed time across phases, check between passes, and use a process watchdog for a stalled pass. On expiry, remove hooks where possible, preserve completed and partial records, and report a bounded incomplete run. Do not perform generation, optimization, extra layer searches, or ad hoc prompt retries.

4 Part 3 — Save exact fixtures before observing results

The following are original synthetic fixtures. The plus/minus labels describe the human-authored context, not a verified model representation. Use every character exactly as shown. In JSON, `\n` means one newline. Append the same suffix to every review context: `\nOverall, it was.`

```
{
  "discovery": [
    {"id": "D1+", "context": "The meal was delicious and the service was excellent."},
    {"id": "D1-", "context": "The meal was disgusting and the service was terrible."},
    {"id": "D2+", "context": "The room was comfortable and the staff were helpful."},
    {"id": "D2-", "context": "The room was uncomfortable and the staff were rude."},
    {"id": "D3+", "context": "The film was delightful and the story was engaging."},
    {"id": "D3-", "context": "The film was dreadful and the story was boring."}
  ],
  "heldout_reviews": [
    {"id": "H1+", "context": "The concert was wonderful and the musicians were talented."},
    {"id": "H1-", "context": "The concert was awful and the musicians were unskilled."},
    {"id": "H2+", "context": "The journey was pleasant and the seats were spacious."},
    {"id": "H2-", "context": "The journey was unpleasant and the seats were cramped."}
  ]
}
```

```

],
"heldout_offtask": [
  {"id": "01", "prompt": "One plus one equals"},
  {"id": "02", "prompt": "The capital of France is"}
],
"review_suffix": "\nOverall, it was",
"score_tokens": [" good", " bad"]
}

```

Use no chat template, no additional beginning-of-sequence token, no padding, and no truncation: `add_special_tokens=False`. Process prompts individually. Assert each sequence contains between 1 and 64 tokens. Save text, token IDs, decoded token pieces, and zero-based positions. Assert that the final token ID is the same for all ten review prompts. The off-task prompts deliberately have different final tokens.

Verify that " good" and " bad", each including its leading space, encode as one token apiece and have different IDs. If either is multi-token, stop; do not silently change the metric or use only its first token. No token ID or observed norm is supplied as an expected answer.

It is permissible to validate held-out tokenization before freezing predictions. Do not run the held-out prompts through the model, view their activations, or inspect their logits before the freeze. Keep their outputs in a separate phase.

Matching the review suffix controls the final token and wording there. It does not match prefix lengths, token positions, adjectives, frequency, or subject matter. List these confounders before running anything.

5 Part 4 — Identify and validate the exact boundary

Use zero-indexed block 2, the third of six blocks, for both extraction and intervention. There is no layer search. Hook `model.gpt_neox.layers[2]` after its forward method returns. In Transformers 4.57.1's `GPTNeoXLayer`, the result is a tuple whose first element is the raw post-block residual tensor. Assert that type and the shape `(1, n, 128)` at runtime. Do not confuse this class with another decoder-layer implementation in the same source file.

Select `[0, n-1, :]`: the final input position. The intervention is after both block-2 branch contributions and the skip input have been summed, before block 3 starts. It is not an intervention in an MLP hidden unit or in the model's final `LayerNorm` output.

An observation hook clones and detaches the selected vector, increments a per-run call counter, and returns `None`. It must be called exactly once. An intervention hook clones the complete residual tensor, edits only that final row, and returns `(replacement,) + output[1:]`. Never mutate the original output in place. Assert that every earlier row remains exactly unchanged at the hook boundary. Save copies of the original and replacement final rows.

Use one context-managed hook per pass, with handle removal in `finally`. Compare module hook registrations before and after each context. Keep no global or permanent hook. PyTorch's forward-hook documentation explains removable handles and output replacement.

5.1 Required checks, using discovery inputs only

1. Run all six discovery prompts with no hooks; save final-position logits.
2. Run all six again with observation-only capture. For each, report the maximum absolute difference from its uninstrumented logits. Require `torch.allclose` with `atol=1e-6`, `rtol=1e-6`, and report actual maximum error even if it passes.
3. Repeat unhooked D1+; check against its original baseline.
4. Run D1+ through a zero-addition hook using an explicit length-128 zero vector. Check its logits against baseline and confirm zero change at the target row.
5. Register a temporary hook on block 2 that raises a deliberately named exception on its one invocation. Catch only that expected exception outside the hook context. Verify the handle is removed despite the failure.
6. Run unhooked D1+ again and compare with baseline. Assert all runner-owned hooks have been removed.

These steps consume sixteen attempted forwards: six baseline, six observer, and four additional checks. A failure is a stop condition. Fix instrumentation before making any causal interpretation; retain failed-run logs separately rather than overwriting them.

6 Part 5 — Construct the direction using discovery only

Let r_k^+ and r_k^- be the captured final-position vectors for discovery pair k . Calculate, using float64 analysis of the captured float32 values,

$$d = \frac{1}{3} \sum_{k=1}^3 (r_k^+ - r_k^-), \quad u = d / \|d\|_2, \quad R = \frac{1}{6} \sum_{k=1}^3 \left(\|r_k^+\|_2 + \|r_k^-\|_2 \right).$$

Require finite values and $\|d\|_2 > 10^{-8}$ and $R > 10^{-8}$. Otherwise report a degenerate direction and stop. Do not substitute a favorable coordinate or another layer. Set $\delta = 0.05R$. Save d , its norm, u , R , and δ . The constants 0.05 and 0.10 below are chosen experimental budgets, not established safe or optimal steering strengths.

Create one random comparison direction with a dedicated CPU `torch.Generator().manual_seed(19)`: draw a 128-entry float64 standard-normal vector and normalize it to q . Save its raw draw and normalized values. Do not redraw if the direction produces an inconvenient result. Report $u^T q$; do not force orthogonality. Cast intervention vectors to float32 only when writing them into the model, recording the actual cast-vector norms.

The positive orientation remains positive-context minus negative-context. Do not flip it to make the score improve. Apply $+\delta u$ once to each of the six discovery prompts, using the verified hook. Save these six pilot results, including all disagreements with the intended positive direction. Total so far: twenty-two attempted forwards.

6.1 Freeze predictions before the held-out pass

The primary hypothesis is that adding $+\delta u$ produces a positive **mean within-prompt change across the four held-out reviews** in the final-position logit difference

$$y = z_{\text{good}} - z_{\text{bad}}$$

on those reviews. The opposite-sign hypothesis is a negative mean within-prompt change for $-\delta u$ over those same four reviews. Individual sign counts are consistency diagnostics, not the primary endpoint. Keep these predeclared signed hypotheses as the hypotheses being tested even if the discovery pilot weakens them. Separately record your honest expected outcome and confidence after seeing discovery results; you may expect the primary hypothesis to fail. Also record whether you expect the two-times dose to strengthen the effect or become less consistent, and why. A fixed test hypothesis and your updated belief are different records, neither of which the analysis should rewrite afterward.

For projection removal, predict a sign or explicitly predict mixed signs with an explanation. Removing a signed, prompt-dependent projection does not imply the same change as subtracting δu .

Save `predictions.json` with the checkpoint, layer, boundary, fixture hash, direction and random-vector hashes, doses, metric, signed primary/opposite hypotheses with their fixed four-review mean endpoint, your updated expected outcome and confidence, diagnostic predictions, controls, and the discovery pilot summary. Add a timestamp and SHA-256. The held-out phase must require that exact file and verify its hashes. No held-out information may change its contents.

7 Part 6 — Run all held-out arms

For each held-out prompt, in fixed order H1+, H1-, H2+, H2-, 01, 02, run these nine conditions in the stated order. Reset to the unchanged model and input every time.

Condition	Replacement of final-position raw residual r
Baseline	No hook
Zero control	$r + 0$ using the intervention machinery
Positive small	$r + \delta u$
Negative small	$r - \delta u$
Positive larger	$r + 2\delta u$
Negative larger	$r - 2\delta u$
Random positive	$r + \delta q$
Random negative	$r - \delta q$
Direction removal	$r - (u^T r)u$

The random additions are matched-norm controls for the small additions, within declared float32 tolerance. They do not provide a matched-norm comparison for the larger additions or variable-sized projection removal. Label that limitation explicitly. One random direction is a comparison, not a null distribution or a significance test.

Projection removal uses the frozen discovery u but the current prompt's original r . Compute its scalar projection in float64, form the replacement, then cast to float32. Do not recompute u from held-out prompts. The removed component may be much larger than δ ; record its actual perturbation norm and ratio to $\|r\|$. Never describe this arm as an equal-dose comparison.

The six prompts and nine arms consume fifty-four forwards. Finally, rerun all six unhooked held-out baselines and compare with their earlier baselines. Total planned count: **82 attempted forwards**, including the deliberately aborted check. The hard cap remains 90. Additional experiments require a new, clearly exploratory plan rather than an unnoticed sweep.

8 Part 7 — Measure the intended effect and collateral change

For every arm, require all captured and replacement vectors, logits, and derived metrics to be finite. A non-finite value stops the experiment; preserve partial records and identify the failing arm instead of dropping its row. For every completed pass save final-position logits as float32 arrays. Compute analysis metrics in float64 from those saved values. Record:

- y and the within-prompt difference $\Delta y = y_{\text{arm}} - y_{\text{baseline}}$.
- Full-vocabulary probabilities of the two score tokens, their logits, the greedy next-token ID, and its decoded piece. There is no sampled continuation in this Lab.
- $\|r\|_2$, original and replacement projections onto u , actual perturbation norm, and relative perturbation norm. Baselines may reuse their verified zero-control capture for these internal values, labeled as such.
- Full-vocabulary $D_{\text{KL}}(p_0 \| p_{\text{arm}}) = \sum_v p_0(v) [\log p_0(v) - \log p_{\text{arm}}(v)]$, with natural logs and `log_softmax`. Do not compute KL after restricting to the two score tokens.
- Maximum absolute baseline-repeat error, zero-control error, hook invocation count, and hook-cleanup status.

For the four held-out reviews report every Δy , the mean, minimum, maximum, and number with the predicted sign. Use a descriptive near-zero band of $|\Delta y| \leq 10^{-5}$ and keep the unrounded measurements. This band is a reporting convention, not statistical significance or proof that smaller effects are nonexistent. Report the two review pairs separately as well; four rows are not four independent semantic domains.

Keep the two off-task prompts separate from the primary review mean. Their score-token difference is still computable but is not task success. Their KL and greedy-token changes are crude collateral-effect checks. Two unchanged prompts cannot establish preserved general capabilities, and a changed greedy token cannot by itself establish loss of understanding.

Compare small positive and negative effects. Are they opposite in sign, similar in magnitude, or asymmetric? Does the larger dose help, reverse, or mainly disturb the full distribution? Compare both random signs, including the strongest random effect rather than only a convenient one. Do not exclude rows because the baseline preferred the unexpected adjective.

9 Part 8 — Implement a reproducible runner

The implementation should support the following command contract; these are commands for the program you build, not a claim that a supplied script already exists:

```
python run_lab19.py discover --model-dir LOCAL_CACHE --out RUN_DIRECTORY
# Inspect discovery/pilot results, then write PREDICTIONS_FILE.
python run_lab19.py freeze --run RUN_DIRECTORY --expectations-json
```

```
PREDICTIONS_FILE
python run_lab19.py evaluate --model-dir LOCAL_CACHE --run RUN_DIRECTORY
python run_lab19.py summarize --run RUN_DIRECTORY
```

discover performs token validation, the sixteen instrumentation attempts, direction construction, and the six pilot passes, then saves a plan draft without executing held-out prompts. Inspect those results and write your predictions in PREDICTIONS_FILE. freeze records those post-discovery predictions and their hashes before permitting evaluation; it performs no model forwards. evaluate refuses absent, modified, or mismatched prediction files, performs the held-out arms and recovery baselines, and appends an immutable chronological attempt log. summarize loads saved outputs without model forwards and cannot revise predictions. Carry the forward counter across phases.

Required outputs are: environment.json, artifact-manifest.json, fixtures.json, tokens.json, discovery-vectors.npz, predictions.json, attempts.jsonl, metrics.csv, logits.npz, checks.json, and report.md. Use NumPy archives or another transparent non-pickle array format; do not require unsafe pickle loading to inspect results. Save the runner's SHA-256. Refuse accidental overwriting of an existing run directory.

The attempt log must identify the phase, prompt, arm, attempt number, start/end status, expected versus unexpected exception, and hook cleanup. A failed assertion must leave an intelligible partial record, not a fabricated completed row. Store full results even when the headline mean is zero or points in the wrong direction.

10 Explain the result

Write 300–500 words addressing:

1. What precisely was changed, and what was measured?
2. Did observer invariance, zero controls, and restoration baselines pass?
3. What was the sign or near-zero status of the primary held-out mean effect? Separately, were the individual prompt effects consistent or mixed?
4. How did matched-norm random changes compare? What does one random vector leave unresolved?
5. Which lexical, positional, metric, sample-size, and distribution-shift limitations remain?
6. What narrower causal statement is justified, and what stronger semantic statement is not?

A suitable conclusion might establish only that a particular perturbation changed a specified logit contrast in this checkpoint. Do not claim discovery of a universal sentiment feature, reproduction of Llama-2 CAA performance, subjective emotion, or a general safety control.

i Analytical warm-up checks

For $x = [2, 1]^T$, the baseline score is 4. Adding u gives 5, subtracting u gives 3, and projection removal gives 2. The perturbation norms are 1, 1, and 2 respectively.

For $x = [-2, 1]^T$, the baseline score is 0. Adding and subtracting give 1 and -1 . Projection removal again produces $[0, 1]^T$ and score 2, now by adding $2u$. Removing a projection depends on its signed coefficient.

No trained-model results are prescribed. A correct implementation with unsuccessful steering is a valid completed Lab.

11 More Learning

- Steering Llama 2 via Contrastive Activation Addition, v4: compare its answer-conditioned data and intervention positions with this small prefix-only experiment.
- Towards Best Practices of Activation Patching, v2: examine how metric and corruption choices can alter conclusions.
- Return to Lesson 5.3 and identify the difference between a causal effect of this edit and a complete causal explanation of the model's ordinary behavior.

12 Optional implementation files

The reproducible reference uses `run_lab19.py`, shared contrast experiment, exact Lab 19 fixtures, offline runtime helper, and pinned artifact verifier. Save these files together and use the existing pinned CPU environment; the commands above use the matching runner filename. Generated evidence stays in a learner-owned directory. This reference does not supply an expected trained-model result.

This reference adds a mandatory post-discovery freeze step: inspect the saved discovery measurements and pilot, write your predictions in a separate JSON file, then run `python run_lab19.py freeze --run RUN_DIRECTORY --expectations-json PREDICTIONS_FILE` before evaluate. Required nonempty text fields are `expected_outcome`, `confidence`, `larger_dose`, and `projection_removal`. Discovery cannot preload or automatically choose those predictions; the freeze uses no model forwards.