The Negotiated AI Training Loop

The **AI Training** desk gives an operator one place to propose, run, monitor, evaluate, and learn from model-training experiments—without handling GPU infrastructure or credentials directly.

The operator's training loop

1. Open AI Training

From the application menu, open the AI Training desk at `/training`.

The page has three complementary areas:

- **ML Control Plane runs:** current and previous experiments.
- **New drafting experiment:** the form for starting a run.
- **Training Review queue:** document-level results requiring human judgment.

2. State the hypothesis

Select **New MLCP run** and describe the experiment before spending GPU time.

The operator chooses:

- Base model
- Number of training epochs
- LoRA rank
- Approved training dataset
- Maximum training time
- The existing model or adapter used as the reference
- The metric expected to improve
- The minimum improvement that would count as meaningful
- A plain-language description of what is changing and why

This forces every run to answer: **What are we testing, what should improve, and what are we comparing it with?**

3. Submit the experiment

Choose **Submit to ML Control Plane**.

Negotiated securely sends the experiment to MLCP. The browser never receives MLCP credentials, RunPod credentials, Hugging Face credentials, or artifact-store credentials.

Before submitting, Negotiated saves the run identity and exact request. If the network drops after MLCP accepted the run, the operator can safely retry the same submission without accidentally creating and billing for a duplicate experiment.

4. Monitor the run

The **ML Control Plane runs** table shows:

- Run ID
- Current status
- Base model
- Hypothesis
- Accumulated cost
- Last update time

Statuses move through states such as queued, claimed, running, succeeded, or failed.

The operator no longer needs to inspect the local inference-job queue to determine whether training is running. MLCP is the source of truth.

5. Open the run details

Selecting a run opens its dedicated page.

That page shows:

- Purpose and status
- Base model
- Training recipe
- GPU time used
- Last update
- Complete hypothesis
- Expected minimum effect
- Reference model
- Reported training and evaluation metrics

If no metrics exist yet, the page says so instead of displaying misleading zeroes.

6. Evaluate the result

The operator compares the reported metrics with the original hypothesis:

- Did the target metric improve?
- Was the improvement large enough?
- Did important guard metrics regress?
- Did the run finish within its budget?
- Is there enough evidence to trust the result?

The interface preserves **not measured**, **too little evidence**, and **unreadable** as distinct states. It does not turn missing evidence into a passing score.

7. Review document-level failures

The **Training Review queue** remains available for detailed exam evidence.

Reviewers can inspect documents worst-first and compare:

- Counterparty language
- The firm's actual revision
- The model's answer
- The substantive verdict
- Separate diagnostic dimensions
- Skipped or unparseable edits

This helps reviewers focus on examples that can teach the model something, rather than spending time on documents it already handled correctly.

8. Correct the model's output

On a document page, a reviewer can write the language the model should have produced.

Corrections remain distinct from the original gold text and the model's failed output. The system records that a person supplied the corrected answer.

Before a correction can become training data, its project must be deliberately retired from the holdout set. This prevents the evaluation set from silently leaking into training.

9. Run the next experiment

Once corrections or dataset changes are approved:

1. Create a new hypothesis.
2. Select the prior champion as the reference.

3. Submit another MLCP run.
4. Monitor cost and execution.
5. Compare its metrics and document-level behavior.
6. Promote only when the evidence supports it.

That closes the iterative loop:

> **Hypothesis → Train → Score → Review failures → Correct → Update data → Train again**

Important current boundary

The MLCP run list, submission, status, cost, and aggregate metrics are connected.

The document-level exam projection and Training Review queue are preserved, but this stack does not yet automatically import every new MLCP run's per-document results into those local review tables. That final projection and synchronization step is needed before a newly completed MLCP run automatically appears as document-level work in the review queue.

The operational loop is therefore connected through execution and aggregate evaluation. Fully automatic handoff from MLCP results into document-by-document correction is the remaining integration seam.