

✦ PLAYBOOK · AI ENGINEERING

Fine-Tuning When It Pays

When fine-tuning beats prompting and retrieval, how to prepare the data, which method to use, and what it really costs to serve and maintain.

FORMAT	LEVEL	READ TIME	UPDATED
Playbook	Advanced	15 min	Oct 2026

OVERVIEW

What this playbook covers

Fine-tuning is often reached for too early and occasionally too late. It is excellent at teaching a model a consistent behavior, format or style, and at letting a smaller model do a narrow job a larger one did before. It is poor at teaching facts that change. This playbook helps you decide when it pays, prepare data that actually improves the model, choose between supervised, preference and parameter-efficient methods, evaluate honestly, and plan for the serving and maintenance cost that follows.

KEY TAKEAWAYS

- Exhaust prompting and retrieval first; fine-tune when you need consistent behavior they cannot deliver.
- Fine-tuning teaches form and behavior well and changing facts badly — keep knowledge in retrieval.
- A few hundred to a few thousand clean, consistent examples usually matter more than a large noisy set.
- Parameter-efficient methods such as LoRA cut training cost and let one base model serve many adapters.
- A fine-tuned model is a product you maintain: it needs evals, retraining and a plan for base-model upgrades.

WHO IT'S FOR

- ML and AI engineers
- Heads of engineering weighing build costs
- Product leads with high-volume LLM features
- Platform teams running model infrastructure

HOW TO USE IT

Use section 01 to decide whether fine-tuning is the right tool at all. If it is, sections 02–05 are the build plan and section 06 is the budget conversation.

CONTENTS

01 Prompt, retrieve or fine-tune?
02 Preparing training data
03 Choosing a tuning method
04 Evaluating a tuned model honestly
05 Serving costs and trade-offs
06 Maintenance and anti-patterns

01 / DECISION

Prompt, retrieve or fine-tune?

These are not rivals. Each fixes a different kind of gap, and most production systems use more than one.

THE GAP	BEST FIRST TOOL	WHY
Model does not know your facts	Retrieval (RAG)	Facts change; retrieval updates without retraining and can cite sources
Model misunderstands the task	Prompting + examples	Cheapest to iterate; often enough on capable models
Output format or style is inconsistent	Fine-tuning	Training bakes the pattern in more reliably than instructions
Large model is too slow or costly at volume	Fine-tune a smaller model	A small tuned model can match a large one on a narrow task
Domain jargon or notation is misread	Fine-tuning, plus retrieval	Training adapts to the language; retrieval supplies the facts

Fine-tuning is worth considering when...

- ☐ A well-engineered prompt with examples still fails on a measurable share of cases
- ☐ The task is narrow, stable and high-volume
- ☐ You can collect hundreds or more high-quality labeled examples
- ☐ You have an eval set that can tell whether the tuned model is better
- ☐ Someone will own retraining and monitoring after launch

02 / DATA

Preparing training data

The model learns exactly what the data shows, including its mistakes and inconsistencies. Data preparation is most of the work.

01 Collect real inputs → 02 Write or curate ideal outputs → 03 Deduplicate and clean → 04 Review for consistency → 05 Split train / validation / test

01

Match production inputs

Train on the messy, real inputs the model will see, not cleaned-up versions written for training.

02

One style, one standard

If labelers disagree on what a good output looks like, the model will learn the disagreement. Write a labeling guide first.

03

Include the hard cases

Ambiguous inputs, refusals and edge cases need examples, or the model will handle them by guesswork.

04

Keep the test set sacred

Hold out an evaluation set before training and never let it leak into training data or prompt examples.

05

Scrub sensitive data

Remove personal data and secrets you would not want the model to reproduce verbatim.

06

Use synthetic data carefully

Model-generated examples can fill gaps, but review them — they carry the generating model's habits and errors.

✦ 03 / METHODS

Choosing a tuning method

Supervised fine-tuning teaches by example; preference tuning teaches by comparison; parameter-efficient methods make either cheaper. Hosted APIs expose some of these; open-weight models give you all of them.

SFT

Supervised fine-tuning

Train on input → ideal output pairs. The default starting point for format, style and task behavior.

PREFERENCE

DPO and similar

Train on pairs of a preferred and a rejected output. Useful for tone and judgment that is easier to compare than to write.

RL

Reinforcement fine-tuning

Optimize against a grader or reward. Powerful for verifiable tasks; needs a reliable grader and more care.

PEFT

LoRA adapters

Train small low-rank matrices instead of all weights. Much cheaper to train and store; quality is often close to full tuning.

PEFT

QLoRA

LoRA on a quantized base model, so larger models fit in less GPU memory during training.

FULL

Full fine-tuning

Updates every weight. Highest ceiling and highest cost; rarely the right first step.

HOSTED FINE-TUNING API
No GPUs to manage
Limited methods and hyperparameters
Weights stay with the provider
Usage-based serving price

OPEN-WEIGHT MODEL YOU TUNE
Full choice of method and model
You own and host the weights
Data can stay in your environment
You run training and serving infrastructure

04 / EVALUATION

Evaluating a tuned model honestly

A lower training loss does not mean a better product. Compare against the strongest baseline you have, on the held-out set, with the metrics the business cares about.

01 **Set the baseline**

Score the best prompted model — with retrieval if you use it — on the held-out set first.

02 **Score the tuned model on the same set**

Same inputs, same graders, same rubric. Report task metrics, not only loss.

03 **Check for regressions**

Test general abilities and safety behavior the task did not cover; tuning can erode them.

04 **Check for memorization**

Probe whether the model reproduces training examples or sensitive strings verbatim.

05 **Review samples by hand**

Read outputs side by side with the baseline. Metrics miss tone and subtle errors.

06 **Decide with cost included**

Weigh quality gains against training, serving and maintenance cost before shipping.

FIELD NOTE

Watch for overfitting to the format

A tuned model can learn the output shape so well that it produces confident, well-formatted answers to inputs it does not understand. Include out-of-scope inputs in the test set.

✦ 05 / COST

Serving costs and trade-offs

Training is a one-off; serving is every day. Model the full cost before committing.

COST LINE	WHAT DRIVES IT	HOW TO REDUCE IT
Training runs	Model size, data volume, epochs, experiments	Start with LoRA on a smaller model
Hosted inference	Provider pricing for tuned models	Compare against base-model price with a longer prompt
Self-hosted inference	GPU hours, utilization, idle capacity	Serve multiple LoRA adapters on one base model
Data labeling	Expert time per example	Label the hard cases; let review cover the rest
Retraining	Data drift and base-model upgrades	Automate the pipeline from day one

One frequent win is replacing a long prompt full of examples with a tuned model that needs only a short instruction. Fewer input tokens per call can offset the higher per-token price of a tuned model at high volume — but measure it on your own traffic rather than assuming it.

✦ 06 / OPERATIONS

Maintenance and anti-patterns

A tuned model drifts out of date as your data, users and base models change. Plan its lifecycle before you ship it.

MAINTENANCE

- ☐ Training data, code and config versioned together
- ☐ Eval suite rerun on every retrain
- ☐ Production samples reviewed and fed into the next dataset
- ☐ Plan for migrating when the base model is deprecated

ANTI-PATTERNS

- [ ] Fine-tuning to teach facts that change monthly
- [ ] Training on outputs nobody reviewed
- [ ] No baseline comparison against a good prompt
- [ ] Shipping on training loss alone
- [ ] Treating the tuned model as finished

> **Fine-tune for behavior, retrieve for knowledge, and measure both against the simplest thing that could work.**
>
> SPLENDENSLABS ENGINEERING PRINCIPLE

GET IN TOUCH

Let's Talk

We help teams decide whether fine-tuning will pay off, then build the data pipeline, training runs and evals to prove it.

- Prompt vs RAG vs fine-tune assessment
- Training data design and labeling guides
- LoRA training and evaluation on your task
- Serving cost model and deployment plan

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem