GO UNDERNEATH

Fine-Tuning, and When Not To

The case against fine-tuning first, then how to do it properly.

First edition, September 2026

Addaly

Series editor: Dr Mustafa Mun

Draft, open for academic review

GO UNDERNEATH

Fine-Tuning, and When Not To

The case against fine-tuning first, then how to do it properly.

Series editor: Dr Mustafa Mun

Addaly

First edition, September 2026 · Draft, open for academic review

Fine-Tuning, and When Not To

First edition, September 2026, build 62a26075.

Written by AI models to a written standard, with Dr Mustafa Mun as series editor. Exam keys checked blind by a second model: 3,665 of 3,666 agreed. Academic review invited.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

How to cite: *Mun, M. (2026). Fine-Tuning, and When Not To. Addaly. addaly.com/learn/fine-tuning.*

This book is the printed form of the course of the same name at addaly.com/learn/fine-tuning. The website is the living version and is corrected first. Where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. The answers to every check, test and examination question are at the back.

9 modules · 84 lessons · 30 figures · 9 case studies · 61,811 words · about 13 hours

Brief contents

1

MODULE 1

Deciding whether to fine-tune at all

Diagnose a model failure as a format, knowledge, style, capability or cost problem, name the cheapest lever that fixes that class, and defend in writing a decision to fine-tune — or not to — against the prompt, retrieval, encoder and distillation alternatives and their licence constraints.

Lesson 1 · 7 min

The Honest Answer First

THE ONE THING TO KEEP

A prompt changes in ten seconds; a fine-tune changes in a day. Exhaust the fast loop first.

Start with the honest answer

Most projects that reach for fine-tuning are solved by a better prompt or by retrieval. Not some. Most. This is the first lesson because if the course changes your mind here, it has done its job and you can stop reading.

Fine-tuning has a slow loop. A prompt change takes ten seconds and you see the result immediately. A fine-tune takes a dataset, a training run, and an evaluation — a day if you are practised, a week if you are not — and when the result is worse, you often cannot tell which of the twenty things you changed did it. Anything you can learn from a prompt, learn from a prompt.

The ladder

Work down this list. Stop as soon as the problem is solved.

1. **Write the instruction properly.** State the output shape. State the constraints. Most "the model won't follow my format" complaints are a prompt that never specified the format precisely.
2. **Put three to five real examples in the prompt.** Few-shot moves format and tone further than people expect, and you can change it at 3am without a GPU.
3. **Give it the facts.** If the answer depends on your documents, policies, or prices, put the relevant text in the context window. This is the answer to "it doesn't know our stuff" nearly every time.
4. **Give it tools.** If the answer depends on a calculation or a live lookup, let it call something.

5. **Try a stronger base model** with everything above, and compare cost honestly.
6. **Now consider fine-tuning.**

The twenty-example test

Before you build anything, do this. Take twenty real inputs your system got wrong — real ones, from logs, not ones you invented. Hand-write the output you wanted for each. Put five into the prompt as examples and run the other fifteen.

- If it now gets most of the fifteen right, you do not need to fine-tune. You need that prompt.
- If it fails because it does not know a fact, you need retrieval.
- If it fails because it cannot do the reasoning at all, fine-tuning is unlikely to rescue it either.
- If it gets them right only when the examples are in the prompt, and your prompt has no room for them at your volume or your latency budget, that is a real fine-tuning case.

Those twenty hand-written outputs are also the start of your dataset, so the test is not wasted work either way.

When fine-tuning is genuinely right

- **A behaviour you cannot prompt reliably at scale.** A strict output shape, a house register, a domain convention that takes 2,000 tokens of instructions to describe. Fine-tuning moves that into the weights and your prompt gets short.
- **Cost and latency.** A tuned 7B that matches a large model on your one narrow task can be ten to fifty times cheaper per token, and faster. At a million requests a month that is the whole business case.
- **Privacy or air-gap.** The model has to run on your hardware, so the large hosted one is not an option at any quality.

- **A narrow task with plenty of labelled data.** Extraction, classification, routing. Here a small tuned model beats a large prompted one on accuracy and cost at the same time.

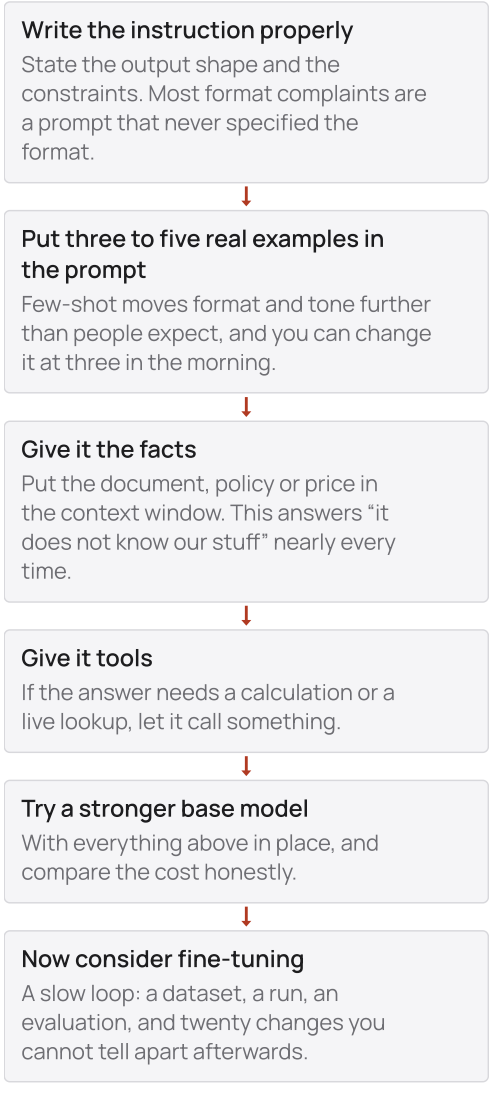
When it is the wrong answer

- Adding facts, especially facts that change. Use retrieval.
- Fixing hallucination. Training on answers the model does not know makes it more confidently wrong, which the next lesson explains.
- Teaching reasoning the base cannot do.
- Anything where you have forty examples, no evaluation set, and a deadline.

The rest of this course assumes you walked the ladder and still have a real case. That is a respectable place to be. It is just a much smaller group than the people currently opening a fine-tuning notebook.

Figure 1.1

The ladder, and where most projects should stop



Work down it and stop at the first rung that solves the problem. A prompt change takes ten seconds; a fine-tune takes a day if you are practised.

BEFORE YOU MOVE ON

A support bot keeps stating your old refund policy, which changed three weeks ago. Your logs show it does this confidently and often. What do you do first?

A. Fine-tune on 2,000 past support transcripts so the model learns how your team handles refunds.

B. Put the current policy text into the context at answer time, then re-check the same failing cases.

C. Fine-tune on 200 examples that state the new policy explicitly.

D. Move to a larger base model, which hallucinates less.

The answer and why: p. 461

Lesson 2 · 8 min

What Fine-Tuning Changes, and What It Cannot Add

THE ONE THING TO KEEP

Fine-tuning teaches the shape of an answer, not its content; facts belong in the context window.

What the gradient actually does

Fine-tuning shows the model an input and a target output, then nudges every trainable weight in the direction that would have made those target tokens slightly more likely. Repeat a few thousand times. That is the entire mechanism, and everything fine-tuning is good and bad at follows from it.

What it moves efficiently is the conditional shape of the output: given an input like this, produce an output like that. Register, length, structure, formatting, when to refuse, when to ask a clarifying question, which of several behaviours

the model already has should fire here. These patterns appear in every example, so every gradient step reinforces them. A hundred examples can visibly change tone. Two thousand can lock a JSON schema so hard the model stops emitting prose at all.

What it does not add: facts

A fact appears in your dataset a handful of times. The base model saw trillions of tokens. To make one fact reliably retrievable you are fighting an enormous prior with a tiny signal, and gradient descent takes the cheaper route: rather than learn the fact, learn the *form* of a confident answer.

That produces the characteristic failure. The tuned model answers in your house style, with your structure, at your length — and invents the specifics. It is worse than before, because it now sounds like your best expert while being wrong.

Even where memorisation does happen, the knowledge is:

- **uncitable** — there is no source to show the user;
- **unupdatable** — the policy changes and you retrain;
- **undeletable** — someone asks you to remove one record and there is no record to remove.

Retrieval has none of those problems, and its failure mode ("I could not find anything") is the one you want.

The hallucination amplifier

Here is the rule that catches most teams. If a training example's answer depends on knowledge the base model does not have, you are training the model to guess confidently.

Say you build 3,000 question-and-answer pairs from your internal wiki and train without showing the model the wiki. Every one of those examples demonstrates the same lesson: when asked about internal matters, produce a specific, confident answer. At inference, asked about something outside the training set, that is exactly what it does.

The fix is not more data. It is to train the behaviour with the context present — examples that include the retrieved passage and an answer grounded in it — or to include examples where the correct output is that it does not have the information.

What it can add, honestly

- **Mapping your language onto what it knows.** Your jargon, abbreviations, ticket codes, product names. That is a translation task and it is learnable.
- **Output shape.** Schemas, tags, field order, the exact vocabulary of a controlled label set.
- **Decision boundaries.** Which of five categories, which team to route to, when to escalate. This is where a small tuned model does best.
- **Refusal and calibration.** Including teaching it to say it does not know, which is a behaviour rather than a fact.
- **Narrow reasoning patterns, partly.** Training on worked traces from a stronger model does raise in-domain performance; that is how many small reasoning models are built. But it transfers the pattern, not the capability underneath. The base's ceiling is still the ceiling, and the gains rarely survive far outside the training distribution.

What it cannot touch at all

Context length. Tool access. Anything about your infrastructure. And behaviour on inputs unlike anything in your data — off-distribution, the tuned model can be worse than the base, which is the subject of a later lesson.

One sentence to keep

If answering requires *knowing* something, put it in the context. If answering requires *deciding* something, fine-tuning can teach the decision.

BEFORE YOU MOVE ON

You fine-tuned a 7B model on 5,000 question-and-answer pairs written from your internal documentation. It now answers in your house style and invents plausible specifics that are wrong, sounding more authoritative than it did before. What happened?

A. The examples taught the form of a confident answer far more efficiently than they taught the facts, so the model now produces the shape of an expert answer with invented content.

B. Two epochs was not enough exposure for the facts to stick; train longer on the same data.

C. The LoRA rank was too low to hold that much knowledge; raise it.

D. The base model is too small; a 70B base would have absorbed the documentation.

The answer and why: p. 462

Lesson 3 · 10 min

Diagnosing the failure before choosing the fix

THE ONE THING TO KEEP

Sort thirty real failures into format, knowledge, style, capability, adherence and cost before choosing a fix; only adherence, style and cost have fine-tuning as their cheapest answer.

Thirty failures, sorted by cause

Nobody fine-tunes because of a hypothesis. They fine-tune because something is wrong and fine-tuning is the most impressive-sounding thing to do about it. The cure is dull: collect thirty real failures, sort them by cause, and count.

CASE STUDY · MODULE 1

The custom model that was already announced

An illustrative case, built from documented patterns.

A twenty-person logistics firm in Pune, whose support inbox handles about nine hundred shipment queries a week.

Ravi ran operations at a freight forwarder in Pune. Customers wrote in asking where a consignment was, why a duty figure had changed, whether a delivery could be rescheduled. A hosted model drafted replies. It was adequate and it was wrong often enough that two people read every draft before it went out.

In March he took a proposal to the two directors: fine-tune a model on eighteen months of the company's own correspondence, so that it would answer "the way we answer". The directors liked it. One of them repeated the phrase "our own AI model" to a customer the following week. A budget of four lakh rupees and six weeks was approved.

The engineer Ravi hired, Sneha, began where the course says to begin. She pulled thirty replies the drafting model had got wrong — real ones, from the archive, not ones anybody invented — and sorted them by what she would have had to change first.

Nineteen of the thirty were knowledge failures. The model did not know the current duty schedule, did not know that one customer had a negotiated rate, did not know which of three warehouses a consignment had been routed through. Six were format: the draft opened with an apology the firm never used, or put the tracking number at the bottom. Three were adherence — the draft ignored two of the four instructions in the prompt. Two were genuine capability failures involving a multi-leg date calculation.

Then she ran the paste test on the nineteen. She put the relevant document into the prompt — the duty circular, the rate card, the routing record — and asked again. Seventeen of the nineteen came back correct.

That is the whole finding, and it was available in a day and a half.

Sneha wrote it up plainly: the firm's problem was that the model could not see its own records. Retrieval over the rate cards, the duty circulars and the shipment database would fix seventeen of thirty failures. Constrained decoding and five examples in the prompt would fix the six format ones. Fine-tuning was the right tool for perhaps three of the thirty, and for those three it would need a dataset, an evaluation set, and somebody to rebuild it every time the base model moved.

She also costed the version Ravi had proposed. Two thousand examples at five minutes each of somebody's careful attention. A held-out set of two hundred, built first. Eight to fifteen training runs rather than one. A card to serve it on, or a provider that hosts adapters. And a standing question every six to nine months, when the base model was superseded, about whether to retrain or to stay frozen at last year's quality.

Ravi read it and saw the difficulty immediately. The technical answer was clear. The organisational one was not. A director had told a customer the firm was building its own model. Coming back with "we are going to write a better prompt and connect the database" sounded, to anyone who had heard the first version, like a retreat.

The cost of the honest answer was a conversation Ravi did not want to have, and the visible shrinking of a project he had proposed. The cost of the other answer was six weeks, four lakh rupees, and a model that would still not know the duty schedule — because no amount of training installs a fact that changes every quarter, and training on those answers would have taught the system to state duty figures confidently whether or not it knew them.

YOUR ANSWERS

1. Sneha spent a day and a half on diagnosis before proposing anything. What did that day and a half actually buy, given that the project had already been approved?

2. Suppose the firm had trained on three thousand question-and-answer pairs drawn from the archive, without retrieval. What specific failure would that have produced, and why is it worse than the original problem?

3. The directors had already told a customer the firm was building its own model. How should a decision record have handled that, and what would it have cost?

STOP HERE

Write your answers before you turn the page. How it played out starts on p. 50.

This page is blank so that how it played out stays out of view until you turn the page.

CASE STUDY · MODULE 1

How it played out

Ravi took the one-page finding to the directors himself, with the thirty sorted failures printed out and the paste-test results beside them. He did not soften it. The retrieval work took nine days, including connecting the shipment database, and moved the first-pass acceptance rate from 61 to 88 per cent. The format work took an afternoon. The three adherence failures were left alone.

The firm never fine-tuned anything. Four months later a new base model arrived that handled the multi-leg date calculation correctly, and the two capability failures disappeared without anybody doing work. The dataset Sneha had started — the seventeen hand-written replies from the paste test — became the few-shot examples in the prompt, and were still there a year on.

One director still describes the system as "our own AI". Ravi has stopped correcting him.

The questions, discussed

1. Sneha spent a day and a half on diagnosis before proposing anything. What did that day and a half actually buy, given that the project had already been approved?

It converted an approved project into a tested hypothesis. Before it, the firm had a complaint — the drafts are wrong — and a fashionable remedy. After it, it had thirty labelled failures, a paste test showing that seventeen of nineteen knowledge failures resolved when the document was in the context window, and therefore a specific prediction about what each remedy would fix. Approval is not evidence, and a budget that has been granted is exactly the situation in which nobody checks.

2. Suppose the firm had trained on three thousand question-and-answer pairs drawn from the archive, without retrieval. What specific failure would that have produced, and why is it worse than the original problem?

It would have produced a hallucination amplifier. Every one of those examples demonstrates the same lesson: when asked about a shipment or a duty figure, produce a specific, confident answer in the house style. Asked about something outside the training set, the model does exactly that — it invents the figure. The result is worse than the starting point because the replies now sound like the firm′s most experienced clerk while being wrong, so the two people reading drafts have lost the cue that used to make them look twice.

3. The directors had already told a customer the firm was building its own model. How should a decision record have handled that, and what would it have cost?

The record should have carried the failure quantified, the diagnosis with its evidence, a frozen baseline and a stated kill criterion, all written before any work started — so that abandoning the fine-tune was a pre-agreed outcome of a test rather than a reversal by one person. Its cost is twenty minutes and a more uncomfortable first meeting, because writing ′we stop if retrieval closes the gap′ in front of the directors forces the smaller version of the project to be visible at the point of approval rather than six weeks later.

MODULE 1 TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record. The answers, and the reason behind each, are in the key on p. 460. If two go wrong, re-read the module before the exam.

1 You take twenty real failures, hand-write the output you wanted for each, put five into the prompt and run the other fifteen. Thirteen now come out right — but at production volume your prompt has no room for the examples. What has the test told you?

A. That you have a real fine-tuning case: the behaviour is promptable and the prompt will not fit.
B. That the base model is too small, and a larger one is the answer.
C. That fifteen items is far too few to conclude anything, so the exercise should be repeated on at least two hundred before any decision is taken.
D. That the model was missing a fact, so retrieval is the lever.

2 A team builds three thousand question-and-answer pairs from an internal wiki and trains on them without ever showing the model the wiki. What does the model learn?

A. It memorises the wiki, but cannot cite it, so the answers are correct and unsourced.
B. Nothing measurable, because three thousand examples cannot move a model of that size.
C. That when asked about internal matters it should produce a specific, confident answer — which it then does for things it does not know.
D. That internal questions should be declined, because the wiki was absent from the context at training time.

3 Your retrieval-augmented system answers 51 per cent of questions correctly. Measuring recall at five on fifty questions with known source passages gives 0.62. What follows?

A. The gap between 51 and 62 shows that retrieval is fine and the remaining errors are reasoning failures that only a larger generator will fix.

B. End-to-end accuracy cannot exceed about 62 per cent until retrieval improves, so the generator is the wrong place to spend.

C. The generator is discarding good evidence, so fine-tuning it on grounded answers is the next step.

D. Recall at five is unrelated to end-to-end accuracy, which depends on the generator alone.

4 A routing task has six fixed queues and two thousand labelled tickets. What does a fine-tuned 150M encoder buy over prompting a large hosted model, and what does it cost?

A. It is more accurate from day one with no labels at all, and costs only a little GPU memory.

B. It gives the same accuracy at the same price, and its only advantage is that the weights are yours.

C. It is less accurate but much cheaper, so it suits situations where the quality of the routing decision does not matter very much to anybody.

D. It is usually more accurate above a thousand labels and costs nothing per call, but it needs labels and a retrain to add a class.

5 A student is distilled from a teacher that is wrong on six per cent of your inputs, with no filter between them. What happens to those six per cent?

A. The student learns them as correct answers, and states them with less hedging than the teacher did.

B. The student rejects them, because a smaller model cannot represent a complicated error.

C. They appear only on inputs outside the distribution the teacher was sampled on, which is where every distillation fails first and where filtering could not have helped.

D. They are diluted by the ninety-four per cent and have no measurable effect on the student.

6 A fine-tune works well, and the base model family ships a meaningful upgrade every six to nine months. Which artefact decides whether that upgrade is an improvement or a threat?

A. The original training notebook, provided it still runs on the current library versions and nobody has changed the rank or the learning rate in it since.

B. The adapter weights, kept in version control with a content hash beside them.

C. The dataset, the configuration and the evaluation set, which are what a rebuild needs.

D. The serving infrastructure, since a new base model means a new deployment.

Lesson 10 · 10 min

One training step, taken apart

THE ONE THING TO KEEP

A training step is forward, cross-entropy against the next token, backward to gradients, optimiser update — and the memory, the masking and every hyperparameter you will set are consequences of those four lines.

Four things happen, in order

Everything in this course is a variation on one step, repeated a few thousand times. The step has four parts: a forward pass, a loss, a backward pass, and an update. If you can narrate these with real shapes, every hyperparameter later stops being folklore.

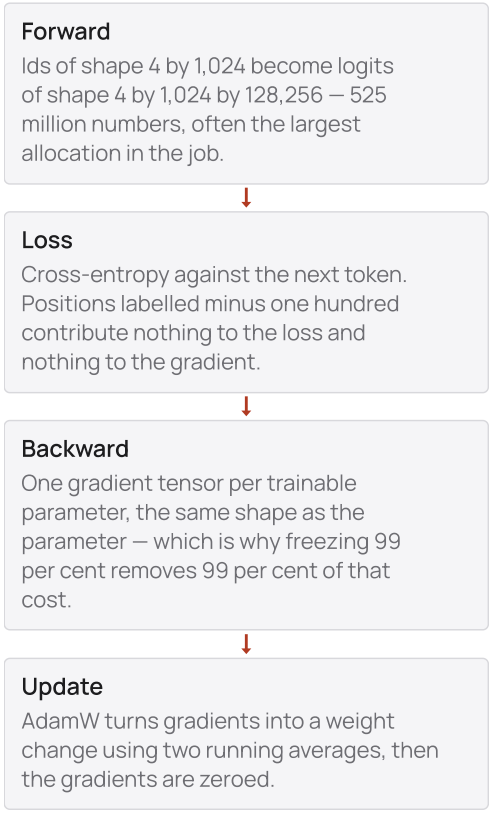
Take a concrete batch: 4 sequences of 1,024 tokens each, on an 8B model with a vocabulary of 128,256.

1. Forward

The token ids go in as a tensor of shape `(4, 1024)`. The embedding table turns each id into a vector of 4,096 numbers, giving `(4, 1024, 4096)`. That tensor passes through every transformer block — attention, then a feed-forward network — and comes out the same shape at the end.

Figure 2.1

One training step, in four parts



A batch of four sequences of 1,024 tokens on an 8B model. Every hyperparameter later in the course is a consequence of one of these four lines.

The last layer projects it to vocabulary size: `(4, 1024, 128256)`. These are the logits, one score per possible next token, at every position.

Notice the size of that tensor. Four times 1,024 times 128,256 is 525 million numbers. In fp32 that is 2.1 GB for a single forward pass, and the backward pass needs room for its gradient too. On many real runs the logits are the largest single allocation in the whole job, bigger than the weights of the layers that produced them. This is why fused or chunked cross-entropy kernels — Liger's, or cut cross-entropy — are worth switching on: they compute the loss without ever materialising the full logits tensor, and they routinely cut peak memory by 30 to 50% on models with large vocabularies.

EXAMINATION

Fine-Tuning, and When Not To — final examination

This paper covers the whole course:

- deciding whether to fine-tune at all
- how a training step works
- the parameter-efficient family and its memory arithmetic
- building a dataset
- the objectives beyond supervised fine-tuning
- the mechanics of a run
- adapting models that are not chat models
- measuring whether it helped
- shipping and maintaining what you built

63 questions, the whole pool. The site draws 25 for a sitting, pass mark 70%. Work any 25 under your own conditions. Answers from page 516.

1 Module 1 · Deciding whether to fine-tune at all

You paste the missing document into the prompt and the model answers correctly. What has that test established?

A. That fine-tuning on document-and-answer pairs would work, since the model has now demonstrated it can use this document correctly when shown it.
B. That the model has the ability and lacked only the information, so the lever is retrieval.
C. That the model's capability is marginal and a larger base is needed.
D. That the failure was a format problem hiding behind a knowledge one.

2 Module 1 · Deciding whether to fine-tune at all

Thirty failures sort mostly into the format bucket. Why try constrained decoding before fine-tuning for format?

A. Because a grammar also improves the content inside the structure.
B. Because a schema enforced at sampling time reduces the number of tokens generated, which is where the saving on a format-heavy task actually comes from.
C. Because fine-tuning cannot change output shape at all.
D. Because it makes invalid output impossible rather than merely rarer, and costs nothing to try.

3 Module 1 · Deciding whether to fine-tune at all

A complaint reads: "it ignores my instruction to be concise". Which cause is this most often, and why?

A. Style, because the model has a strong prior about answer length and your one word is arguing with millions of examples.
B. Instruction adherence, because the model is satisfying some of the constraints in the prompt while silently dropping the one that arrived earliest in it.
C. Capability, because judging the right length is a reasoning task.
D. Knowledge, because the model does not know what your product considers concise.

4 Module 1 · Deciding whether to fine-tune at all

Distillation replaces a $16,000 monthly hosted bill with roughly $500 of self-serving, against a one-off of a few hundred dollars of generation and days of somebody's attention. At what point does that stop being a good deal?

A. When the teacher and student come from different model families.

B. When the task is narrow enough that a prompt already works.

C. At low volume, because the saving is per request and the fixed costs do not amortise.

D. When the student is smaller than about three billion parameters, below which imitation of a larger model stops transferring anything useful at all.

5 Module 1 · Deciding whether to fine-tune at all

Before a fine-tune there are three separate permissions to check. Which three?

A. The model licence, the cloud provider's terms, and your own privacy notice.

B. Copyright, trademark, and your customer contracts.

C. The dataset's own licence, the terms under which its contents were generated, and whether the resulting weights can be exported to another country.

D. The model licence, the data licence, and the obligations that attach to what you produce.

Module 1 · Deciding whether to fine-tune at all

The module starts on p. 11.

MODULE 1 TEST

Module 1 test, Q1, p. 52
You take twenty real failures, hand-write the output you wanted for each, put five into the prompt and run the other fifteen. Thirteen now come…

Answer A: That you have a real fine-tuning case: the behaviour is promptable and the prompt will not fit.

Why: The examples worked, so the behaviour is learnable from demonstrations rather than missing knowledge or absent capability. What fails is delivery: the instruction that produces the behaviour does not fit your context or latency budget at volume. Fine-tuning moves that behaviour into the weights and shortens the prompt, which is precisely the case it is for.

Module 1 test, Q2, p. 52
A team builds three thousand question-and-answer pairs from an internal wiki and trains on them without ever showing the model the wiki. What does the…

Answer C: That when asked about internal matters it should produce a specific, confident answer — which it then does for things it does not know.

Why: A fact appears a handful of times against a pretraining corpus of trillions of tokens, so gradient descent takes the cheaper route and learns the form of a confident answer rather than the fact itself. Every example demonstrates the same lesson — internal question, specific confident reply — and at inference the model applies it to things it was never told.

Module 1 test, Q3, p. 53
Your retrieval-augmented system answers 51 per cent of questions correctly. Measuring recall at five on fifty questions with known source passages gives 0.62. What follows?

Answer B: End-to-end accuracy cannot exceed about 62 per cent until retrieval improves, so the generator is the wrong place to spend.

Why: A passage that was never in the context window cannot be used, so recall at k is a hard ceiling on how often the system can be right for the right reason. Eleven points sit below that ceiling and thirty-eight sit above it. Fine-tuning the generator can only work on the eleven, and training it to answer without the evidence teaches exactly the confident guessing you are trying to stop.

Module 1 test, Q4, p. 53
A routing task has six fixed queues and two thousand labelled tickets. What does a fine-tuned 150M encoder buy over prompting a large hosted model…

Answer D: It is usually more accurate above a thousand labels and costs nothing per call, but it needs labels and a retrain to add a class.

Why: The encoder reads the whole input bidirectionally and is optimised directly for the decision, while the decoder is optimised to continue text and is being asked to decide indirectly. Above roughly a thousand labels the encoder usually wins on accuracy as well as on cost and latency. The trade is real: you need labels, a new class means a retrain, and the output is a label and a probability with no explanation attached.

Module 1 test, Q5, p. 54

A student is distilled from a teacher that is wrong on six per cent of your inputs, with no filter between them. What happens to…

Answer A: The student learns them as correct answers, and states them with less hedging than the teacher did.

Why: Supervised fine-tuning maximises the probability of the answers in front of it and has no way to represent that one of them is wrong, so the teacher's mistakes arrive as gold. The student also learns the teacher's words rather than its uncertainty, so it reproduces those mistakes more flatly. The filter between teacher and student — a verifier, rejection sampling, a hundred pairs read by a person — is where the quality actually comes from.

Module 1 test, Q6, p. 54

A fine-tune works well, and the base model family ships a meaningful upgrade every six to nine months. Which artefact decides whether that upgrade is…

Answer C: The dataset, the configuration and the evaluation set, which are what a rebuild needs.

Why: A LoRA is a set of deltas for one exact set of base weights and does not transfer; the model was always the perishable output. What transfers is the pipeline — the dataset with its filters, the configuration you arrived at, the held-out and canary sets — so a team that can rebuild in an afternoon treats a new base as free improvement, and a team that cannot treats it as a threat.

THE LESSON CHECKS

Lesson 1, p. 16

A support bot keeps stating your old refund policy, which changed three weeks ago. Your logs show it does this confidently and often. What do…

Answer B: Put the current policy text into the context at answer time, then re-check the same failing cases.

Why: A prompt changes in ten seconds; a fine-tune changes in a day.

A Fine-tune on 2,000 past support transcripts…: Those transcripts do teach house behaviour, but they encode the old policy, so you would train the wrong answer in more firmly.

C Fine-tune on 200 examples that state…: It targets the exact gap, but you repeat the whole exercise at the next policy change, and the model still cannot cite a source.

D Move to a larger base model…: Larger models are better calibrated, but no model of any size knows a policy your company wrote three weeks ago.

Lesson 2, p. 19

You fine-tuned a 7B model on 5,000 question-and-answer pairs written from your internal documentation. It now answers in your house style and invents plausible specifics…

Answer A: The examples taught the form of a confident answer far more efficiently than they taught the facts, so the model now produces the shape of an expert answer with invented content.

Why: Fine-tuning teaches the shape of an answer, not its content; facts belong in the context window.

B Two epochs was not enough exposure…: More passes do increase memorisation, but they memorise your exact strings and increase forgetting long before the facts become reliably retrievable.

C The LoRA rank was too low…: Rank genuinely bounds how much the adapter can change, but higher rank buys verbatim memorisation of your sentences rather than knowledge the model can use.

D The base model is too small…: Bigger bases do absorb more from fine-tuning, but the same failure appears at 70B — the fix is putting the document in the context, not enlarging the container.

Lesson 3, p. 23

A support assistant keeps stating the wrong refund window. Someone pastes the refund policy document into the prompt and the model answers correctly every time…

Answer A: A knowledge failure — the fix is getting the document into the context, not changing the weights.

Why: Sort thirty real failures into format, knowledge, style, capability, adherence and cost before choosing a fix; only adherence, style and cost have fine-tuning as their cheapest answer.

B That the base model is too…: Treats private company policy as general knowledge; no model of any size was pretrained on your internal document.

C That the model can learn the…: Confuses reading a fact in context with storing it in weights; the paste test shows the model can use the fact, not that gradient updates will reliably install it.

D That the prompt is too short…: Mistakes the mechanism: what helped was the specific fact being present, not the prompt being longer or more emphatic.

GO UNDERNEATH

Fine-Tuning, and When Not To

The case against fine-tuning first, then how to do it properly.

WHAT IT TEACHES	Fine-tuning is the most over-reached tool in applied machine learning.
WHAT IS INSIDE	Nine modules and 84 lessons, 30 figures drawn from data, nine case studies, nine module tests, a 63-question examination, and every answer with the reason behind it.
LICENCE	CC BY-NC-SA 4.0. Copy, print, share and adapt it for any non-commercial purpose, with credit, under the same licence. There is no paid edition.
THE COURSE	Free to read, with the lessons, the films and the examination, at **addaly.com/learn/fine-tuning**

Scan to open the course

Addaly

First edition, September 2026 · build 62a26075