

3.2 — Post-Training, Alignment, and Safety Learned into Weights

Gavin · AI-assisted course design

Contents

1	From predicting text to acting as an assistant	3
2	Base and assistant describe training roles	3
3	Supervised fine-tuning teaches from demonstrations	4
3.1	An exact supervised-loss calculation	5
4	Preferences supply comparisons rather than complete targets	5
5	RLHF optimizes a policy against feedback	6
5.1	What the regularizer changes	6
6	Direct preference optimization takes a different route	7
7	Reasoning training and reasoning time are different	8
8	Tool-use training does not execute a tool	9
9	Safety can be learned without becoming a guarantee	9
10	Read a checkpoint as an evidence package	10
11	Check your understanding	11
12	More Learning	11

1 From predicting text to acting as an assistant

A model can complete a passage about answering questions without reliably answering the question you just asked. Predicting plausible continuations and being a useful assistant overlap, but they are different targets. A continuation might introduce another speaker, repeat the question, or imitate a document containing mistakes. An assistant is expected to follow the request, use the relevant context, acknowledge uncertainty, and respect constraints.

Lesson 3.1 explained how an objective changes parameters through training. **Post-training** applies further training to an existing checkpoint, often to make particular behaviors more reliable. It can change instruction following, response style, task performance, tool selection, and safety behavior. It need not change the Transformer architecture.

The central question remains: **where does the behavior live?** Some changes persist in learned weights. Others come from the current prompt, decoding settings, tools, or application. A polished answer does not identify which combination produced it.

By the end, you should be able to:

- distinguish a base checkpoint from an assistant-oriented checkpoint;
- trace demonstrations and preferences into different training losses;
- compute a supervised loss and a reference-relative preference margin;
- distinguish reasoning training from reasoning-time computation;
- separate learned tool-use behavior from tool execution and permission;
- state what documentation, outputs, and evaluations actually establish.

Math to know / refresh

Needed now: conditional probability, negative log-likelihood, weighted averages, and the distinction between an objective and its optimization procedure. All logarithms here are natural logarithms, measured in nats.

Useful refresh: $\log(ab) = \log a + \log b$, odds $p/(1-p)$, and the sigmoid $\sigma(z) = 1/(1 + e^{-z})$.

Side trail: policy gradients, PPO, and the derivation connecting KL-regularized reward maximization to DPO. You do not need their full derivations to follow the worked examples.

2 Base and assistant describe training roles

Write a language model's conditional distribution as $\pi_{\theta}(y \mid x)$: the probability of response sequence y given context x , with learned parameters θ . Calling this distribution a **policy** emphasizes choosing actions, here tokens or responses. It does not imply a separate planning module.

A **base model** is a checkpoint intended as a pretrained foundation. An **instruction-tuned** or **assistant** model has additional training aimed at responding appropriately to instructions. These are descriptions of training history and intended use, not universal architectural classes. A base model can answer questions, particularly with suitable examples. An assistant model still generates tokens using a conditional distribution and can still make mistakes.

Do not interpret “base” as an unfiltered view of reality. Corpus selection, filtering, architecture, and optimization already shaped it. Conversely, do not assume post-training merely places a detachable etiquette layer over an unchanged store of abilities. It updates shared parameters and can improve, reduce, or redirect performance.

The original InstructGPT study provides a concrete historical pipeline: supervised demonstrations, a reward model fitted to human comparisons, and policy optimization using PPO. Its main variant also mixed in pretraining updates. Human preference improved on the study’s evaluation distribution, while limitations and some benchmark regressions remained. These findings concern that setup, not every present-day assistant. Ouyang et al., Sections 3–5

“Alignment” names a desired relationship between behavior and intended goals or constraints. It is not one algorithm, an agreed universal value function, or a certificate awarded by completing a training stage.

3 Supervised fine-tuning teaches from demonstrations

Supervised fine-tuning, or SFT, supplies example inputs and desired outputs. An illustrative record might contain:

```
{ "messages": [
  { "role": "user", "content": "Write blue followed by a period." },
  { "role": "assistant", "content": "blue." }
]}
```

This is an original teaching example, not a universal dataset schema. A training program must serialize those roles and strings, tokenize them, decide where examples begin and end, and choose which positions contribute to the loss. A chat template converts structured messages into the token sequence the model receives. Merely adding role names to a JSON object does not train anything.

For a concatenated sequence $s_1, \dots, s_T$, let m_t be one for a supervised target position and zero otherwise. Assuming at least one supervised position, a token-averaged objective for one record is

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{\sum_t m_t} \sum_t m_t \log \pi_\theta(s_t \mid s_{<t}).$$

The causal shift matters: the prediction before token s_t is scored against s_t . With **teacher forcing**, previous target tokens are supplied from the demonstration when computing later predictions. Training does not first have to generate the correct prefix on its own.

An assistant-only mask excludes prompt tokens from direct target loss, but those tokens remain input context. Gradients from assistant predictions can still flow through computations involving the prompt. In a multi-turn conversation, one recipe might supervise every assistant turn; another might supervise only the last. Packing, truncation, stop tokens, and averaging conventions also matter. A reported “SFT loss” is ambiguous without these details.

Demonstrations can teach concise answers, clarification, task procedures, uncertainty, or appropriate refusal. Their provenance and coverage matter: repeated errors or an overrepresented style become

training signals too. Synthetic demonstrations are still demonstrations; their usefulness depends on their quality and selection.

3.1 An exact supervised-loss calculation

For the example above, invent a tokenizer whose complete assistant target is three tokens: blue, ., and an end-of-response token. Mask every prompt position. Suppose their correct-token probabilities under their respective teacher-forced prefixes are

$$\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{2}\right).$$

The probability of the complete target is their product, $1/16$. The total negative log-likelihood is $\log 16$; the mean token loss is

$$-\frac{\log(1/2) + \log(1/4) + \log(1/2)}{3} = \frac{\log 16}{3} \approx 0.924196.$$

Now compare a second hypothetical checkpoint assigning probabilities $(3/4, 1/2, 3/4)$. The complete-target probability is $9/32$ and the mean loss is

$$\frac{\log(32/9)}{3} \approx 0.422837.$$

That checkpoint assigns more probability to this demonstration. We have not shown how many gradient steps produced it, whether greedy decoding returns this string, or whether unseen instructions improve. Other vocabulary probabilities determine greedy choices. These are stipulated distributions and exact arithmetic, not measurements of a real training run.

Notice the end token: reliably finishing can be part of the target. A model that writes the desired word and then continues for a paragraph can fail a formatting task even when its factual content is harmless.

4 Preferences supply comparisons rather than complete targets

Sometimes a reviewer can choose between two responses more easily than write an ideal one. A preference record contains a prompt x , a preferred response y_w , and a less-preferred response y_l . The subscripts mean winner and loser **under the labeling procedure**. They do not mean true and false.

For our prompt, let y_w be blue. and y_l be Blue is a color.. Both are understandable; the former better follows the requested format. A different instruction could justify the opposite preference. Preserve the prompt and rubric with the label.

A learned **reward model** $r_\phi(x, y)$ produces a scalar score. One pairwise model assigns

$$P_\phi(y_w \succ y_l \mid x) = \sigma \left(r_\phi(x, y_w) - r_\phi(x, y_l) \right),$$

and fits comparisons with loss $-\log P_\phi$. This preference model and its maximum-likelihood formulation are used in the DPO paper’s account of reward learning, Rafailov et al., Section 3

For an invented score pair $r_w = \log 3$, $r_l = 0$, the margin is $\log 3$, the modeled preference probability is $3/4$, and the pair loss is

$$-\log(3/4) = \log(4/3) \approx 0.287682.$$

The preference odds are 3 : 1. Adding 10 to both scores leaves the margin and loss unchanged. Doubling both scores changes the margin and therefore the modeled probability. Score differences have meaning inside this model; individual scores are not universal units of quality.

Nor is 3/4 the probability that blue. is factually true or safe in every context. It is a modeled comparison probability. Reviewers can disagree; the rubric can omit something important; the reward model can mispredict even its own labeling population.

5 RLHF optimizes a policy against feedback

In reward-model-based **reinforcement learning from human feedback**, or RLHF, feedback has two distinct uses. Human judgments train the reward predictor. The reward predictor then scores policy-generated responses used to update the language model. A human need not score every response during policy optimization.

At the token level, the current prefix acts as a state and the next token as an action. A completed response is a trajectory. A policy-gradient procedure changes token probabilities according to estimated returns or advantages. PPO is one such procedure with controls on updates; it is not synonymous with the objective, the reward model, or RLHF as a whole.

A useful schematic objective is

$$J(\theta) = \mathbb{E}_x \left[\mathbb{E}_{y \sim \pi_\theta(\cdot | x)} r_\phi(x, y) - \beta D_{\text{KL}}(\pi_\theta(\cdot | x) \parallel \pi_{\text{ref}}(\cdot | x)) \right], \quad \beta > 0.$$

The reference is a fixed comparison policy, often an earlier fine-tuned checkpoint. At a fixed prompt,

$$D_{\text{KL}}(\pi \parallel q) = \sum_y \pi(y) \log \frac{\pi(y)}{q(y)}.$$

This penalizes departure from the reference distribution, not Euclidean movement of the weight arrays. It is asymmetric. The expectation is nonnegative even though an individual sampled log ratio can be negative. The formula assumes the reference gives positive probability wherever the candidate does. Practical implementations estimate sequence or token-level quantities rather than enumerate all possible responses. InstructGPT documents a KL term as well as its additional pretraining-loss variant. Ouyang et al., Section 3.5

5.1 What the regularizer changes

Invent a world with exactly two possible complete responses, A and B. The reference is (1/2, 1/2), and the stipulated rewards are (1, 0). Its objective is $J = 1/2$.

A candidate policy (3/4, 1/4) has expected reward 3/4 and

$$D_{\text{KL}} = \frac{3}{4} \log \frac{3}{2} + \frac{1}{4} \log \frac{1}{2} \approx 0.130812.$$

At $\beta = 1$, its objective is approximately 0.619188, above the reference. At $\beta = 2$, it is approximately 0.488376, below the reference. The reward ranking did not change; the trade-off did. Neither comparison identifies the globally optimal policy.

Reference regularization can restrain distributional change. It cannot make an incomplete reward function equal the user’s actual intent. Imagine an invented sorting assistant rewarded only for returning valid JSON. It could achieve that metric with an empty list while ignoring the requested sorting task. The defect is in the specification, even if optimization succeeds perfectly.

Reward overoptimization has also been measured experimentally. Gao and colleagues found that optimizing a proxy could eventually reduce the score of the stronger reward model used as their synthetic evaluation target. That “gold” model was an experimental stand-in, not direct access to human welfare or truth. Gao et al., Sections 2–3

6 Direct preference optimization takes a different route

Direct Preference Optimization, or DPO, trains the policy from preference pairs without fitting a separate reward model and running a policy-sampling RL loop during that optimization stage. It uses response log probabilities relative to a reference. This does not eliminate human choices, dataset biases, or optimization problems. Its derivation assumes a particular preference model and KL-regularized objective; finite-data neural-network training need not reach their ideal solution. Rafailov et al., Section 4

Define the reference-relative margin

$$\Delta_\theta = \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} - \log \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\text{ref}}(y_l | x)}.$$

The standard per-pair DPO loss is

$$\ell_{\text{DPO}} = -\log \sigma(\beta \Delta_\theta).$$

Here response log probabilities sum conditional token log probabilities; silently replacing them with length-normalized averages changes the objective.

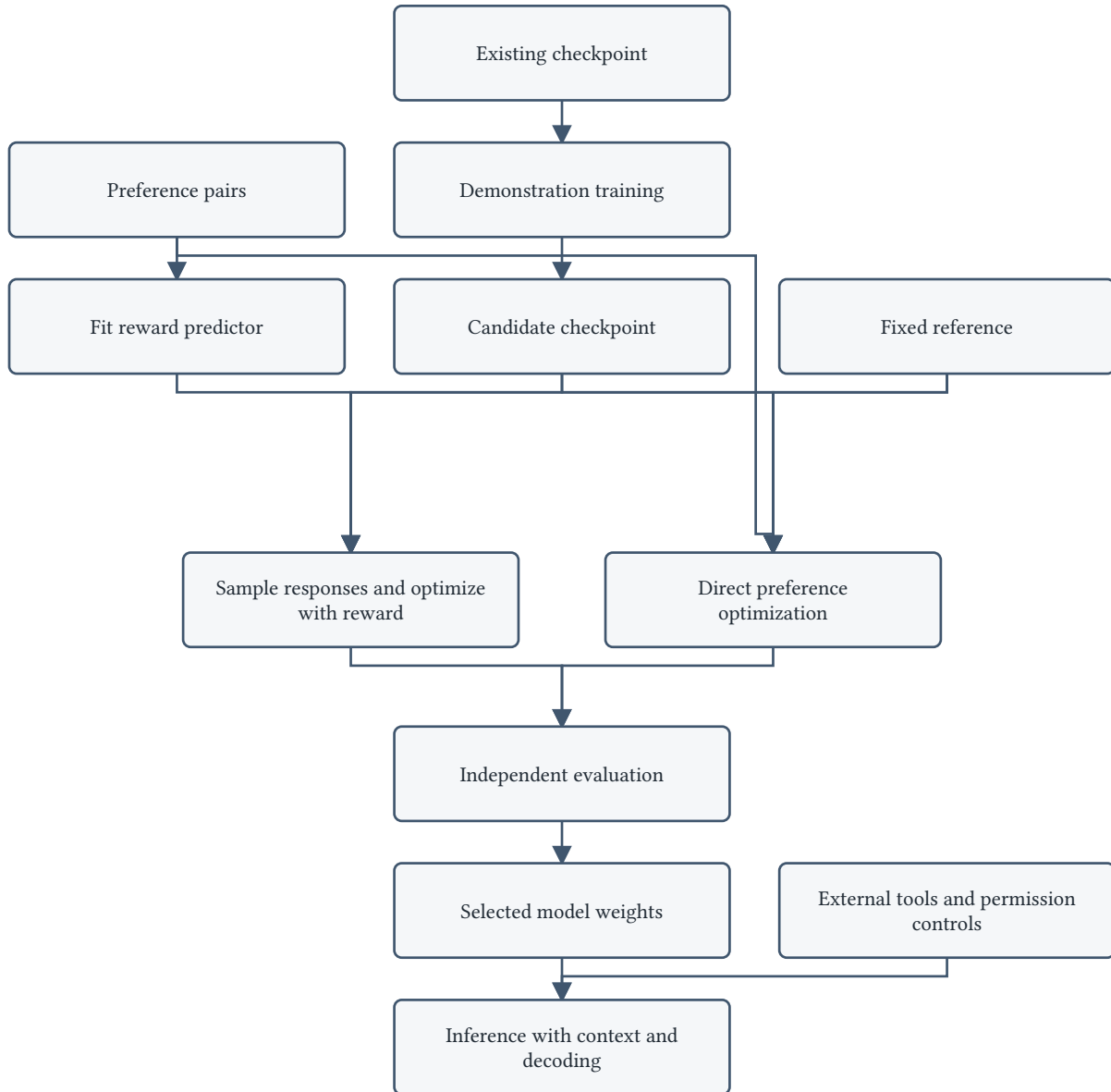
For an original numerical example, assign reference probabilities 0.2 and 0.2 to our two strings. The remaining 0.6 belongs to other responses. Assign candidate probabilities 0.3 and 0.1, leaving the remaining mass unchanged. Reference odds are one, candidate odds are three, and $\Delta_\theta = \log 3$. At $\beta = 1$, the loss is again $\log(4/3)$; at the reference itself, it is $\log 2$.

The same margin results from candidate probabilities 0.15 and 0.05. Both named responses are now less probable than under the reference, yet their relative odds improved by the same factor. A pairwise objective alone does not establish that the preferred response’s absolute probability rose. Other responses and shared parameters matter.

With fixed probabilities, setting $\beta = 1/2$ gives $\sigma(\beta \Delta) = \sqrt{3}/(1 + \sqrt{3})$ and loss about 0.455746. This arithmetic is not a general instruction to maximize β : changing it changes training gradients and the objective’s regularization relationship.

The same β connects to the earlier KL-regularized objective. Under its idealized fixed-reward solution, the optimal reference-relative odds shift is $\Delta^* = (r_w - r_l)/\beta$. A larger β therefore restrains the optimal departure from the reference for that fixed reward problem. This comparison of optima is different from changing β while holding the candidate probabilities fixed in the displayed loss calculation.

SFT, explicit reward-model RL, and DPO can be combined or repeated. Other approaches use AI feedback, verifiable outcomes, or filtered successful samples. Describe the actual data, signal, update, and evaluation rather than treating “aligned” as a complete recipe.



Original course schematic. The two preference branches are alternatives that a larger pipeline may combine. Evaluation selects or rejects candidates; it does not prove universal alignment. External execution controls remain outside the selected language-model weights.

7 Reasoning training and reasoning time are different

Training on worked solutions can change how a model approaches problems. Reinforcement learning can instead score generated attempts using outcomes such as a checked arithmetic answer. These

signals differ: a demonstrated solution supplies target tokens; an outcome score need not specify which intermediate steps were good.

The DeepSeek-R1 report gives a useful contrast. R1-Zero applied RL to a pretrained base without a preliminary SFT stage, using rule-based accuracy and formatting rewards. The R1 pipeline included cold-start demonstrations and multiple subsequent training stages. The report also describes smaller models fine-tuned on generated reasoning data. These are distinct procedures within one report, not interchangeable meanings of “reasoning model.” DeepSeek-AI, Sections 2.2–2.4

At inference, a fixed checkpoint might spend additional computation generating intermediate tokens, sampling several candidates, or checking results with a tool. Its weights need not change. More training compute and more inference compute are separate axes, although training can teach strategies that benefit from a larger inference budget.

For a benign arithmetic task, compare “produce one answer” with “produce three attempts and select a verified answer.” The second system has extra sampling and a selection procedure. If it performs better, that does not isolate a change in model knowledge. Compare equal budgets when the scientific question concerns weights, and report unequal budgets when comparing whole systems.

A written explanation is also not a transparent transcript of neural computation. Experimental interventions on generated reasoning have found task-dependent differences in how much later answers depend on it. Lanham et al., Sections 2 and 5 Judge explanation correctness and behavioral usefulness separately from claims about internal mechanism.

8 Tool-use training does not execute a tool

A training example can teach an assistant to emit a calculator request, read a returned value, and answer. This can improve the choice of tool, arguments, timing, and use of results. Toolformer offers a primary example: candidate API calls were executed and filtered according to their contribution to prediction, then incorporated into further language-model training. Schick et al., Section 2

Consider an invented request to calculate 17×23 . The model emits a structured call naming `multiply` with arguments 17 and 23. An external program must parse the request, check permission, run the function, and return 391. The model then receives that result as context.

The call string alone does no multiplication outside the model. A training transcript containing a tool result does not mean that a deployed model has that tool. A well-formed call does not prove correct arguments, successful execution, or permission. Conversely, an application can provide tools to a model that was not explicitly trained on that exact interface, with variable success.

A safety policy learned in weights may discourage an inappropriate call. A restricted tool API or sandbox can prevent an operation even if the model requests it. These are different protections with different failure modes.

9 Safety can be learned without becoming a guarantee

Demonstrations and preference labels can encourage policy following, uncertainty, boundaries, and helpful alternatives. The resulting response tendencies can persist when the checkpoint is loaded

elsewhere, because the parameters changed. But inference still depends on context and settings; a learned tendency is not a hard execution constraint.

An assistant declining a harmless formatting task could reflect overgeneralized refusal, misunderstanding, insufficient ability, a system instruction, or an external filter. The visible sentence alone cannot distinguish those explanations. Nor does a refusal prove that the model secretly retains a fully functional version of the requested capability. Post-training can change both **what succeeds under a specified evaluation** and **what the model tends to attempt**.

Make capability claims operational: given which prompts, tools, decoding rule, sample budget, and success criterion? A task solved once under one setup establishes something narrower than dependable performance across contexts. Neither “it always knew” nor “the capability was erased” follows from a before-and-after anecdote.

For a toy library assistant, training might encourage answering only from a supplied catalog. A system message can state that instruction anew. A retrieval layer chooses which catalog entries enter context. An application can reject responses lacking an item ID. A permissions layer can prevent any book reservation without approval. All five can contribute to the visible outcome; only the first necessarily describes a weight update.

Robustness must therefore be tested across relevant changes: unfamiliar phrasing, language, longer context, tool failures, and tasks unlike the training set. Benign requests should be included to measure unnecessary refusal. A model that declines everything can score well on a badly designed noncompliance metric while being useless.

10 Read a checkpoint as an evidence package

The public Qwen2.5-0.5B cards provide a manageable example. The base card at revision 060db64 identifies pretraining and advises against using the base directly for conversation. The Instruct card at revision 7ae5576 identifies post-training and names the base in its metadata. Their matching principal dimensions do not imply matching weights.

The family report documents SFT, DPO, and online GRPO stages. Read that as a reported family recipe, with its stated scope; it is not a release manifest tying every training example and optimizer state to a particular file hash. Qwen Team, Section 4

Documentation establishes what authors report. Configuration establishes stored settings. A run establishes observations under its recorded setup. None substitutes for the others. Even a chat template is weak evidence of training history: both the base tokenizer configuration and the Instruct tokenizer configuration contain one. Model names and interface affordances are not provenance records.

 Lab recommended here

Lab 11 — Compare Training Evidence examines the pinned pair’s small public text files, constructs an evidence matrix, and predicts benign behavior. The core requires no model weights, account, paid API, or GPU. An actual generation comparison is a separate optional extension.

For a meaningful evaluation, preserve checkpoint and tokenizer revisions, complete serialized inputs, decoding settings, tool access, sample counts, and scoring rules. Split factual correctness from formatting, usefulness, refusal appropriateness, and tool success. Use held-out examples; inspect possible overlap with training data; report uncertainty and failures rather than only attractive samples. An average benchmark score is evidence for that benchmark and setup, not a universal safety warranty.

11 Check your understanding

1. Does assistant-only loss masking make the prompt irrelevant to training?
2. Why does lower demonstration loss not guarantee a correct greedy answer?
3. What does adding the same constant to two reward scores change?
4. In the two-response KL example, why does doubling β reverse the comparison?
5. Can the DPO margin improve while the preferred string becomes less probable?
6. Does a correct calculator-call string establish successful tool use?
7. What can you infer from a single refusal about training history and preserved ability?

i Answers and reasoning

1. No. The prompt conditions predictions even when its positions supply no direct target loss.
2. The loss scores specified target tokens; greedy output also depends on competitors at each generated prefix.
3. Nothing in their pairwise margin, modeled preference probability, or pair loss.
4. The candidate earns more reward but pays a larger distribution-change penalty. The chosen coefficient changes that trade-off.
5. Yes. Our 0.15 versus 0.05 example improves the odds while lowering both probabilities.
6. No. Check parsing, arguments, permission, execution, result handling, and final answer.
7. Very little by itself. Distinguish hypotheses and obtain documentation or controlled evidence; do not infer an exact recipe or guaranteed hidden capability.

12 More Learning

1. Ouyang et al. — InstructGPT. Read Sections 3.1 and 3.5 for the pipeline, then Section 5 for its scope and limitations.
2. Rafailov et al. — Direct Preference Optimization. Sections 3–4 connect comparison likelihoods, reference policies, and the direct loss.
3. Qwen Team — Qwen2.5 Technical Report. Compare the post-training account with the Lab’s checkpoint-specific files.
4. DeepSeek-AI — DeepSeek-R1. Contrast R1-Zero, R1, and distilled models before generalizing about reasoning training.
5. Schick et al. — Toolformer. Follow how candidate calls become training examples and where actual execution occurs.
6. Gao et al. — Scaling Laws for Reward Model Overoptimization. Keep the synthetic experimental target distinct from true human intent.