Contents

Chapter 6 — Intuitive MCQ Bank (Fine-tuning)

64 multiple-choice questions, intuitive/conceptual level. One correct option each. Cover the answer (blockquote) while testing yourself. Bilingual explanations (English + বাংলা).

Exam style: "Which statement best describes…", exactly one correct option; use full technical terms, no abbreviations.

বাংলা ব্যাখ্যা: এই ব্যাংকে অধ্যায় ৬-এর ৫০+ ধারণামূলক MCQ আছে প্রতিটি প্রশ্নের নিচে উত্তর ও সংক্ষিপ্ত ব্যাখ্যা (ইংরেজি + বাংলা) মুখস্থ নয়, বোঝাপড়া যাচাই করার জন্য বানানো — তাই "কেন", "কী হবে যদি", আর পরিস্থিতি-ভিত্তিক প্রশ্ন বেশি

Topic: Positioning — Where Fine-tuning Fits in the LLM Stack

Q1. Which statement best describes the essential difference between fine-tuning and the techniques of Chapters 3 and 4 (prompting and retrieval)?

- A. Fine-tuning performs gradient updates on the model's parameters, whereas prompting and retrieval steer behavior without touching parameters.
- B. Fine-tuning needs a graphics card, whereas prompting and retrieval can run on any laptop.
- C. Fine-tuning works only on small models, whereas prompting and retrieval scale to large models.
- D. Fine-tuning is always cheaper at inference time, whereas prompting and retrieval are always cheaper to set up.

> **Answer: A.** The chapter frames fine-tuning as the first technique that uses gradient updates to change the weights, while prompting lives in the context window and retrieval in an external index. B confuses hardware with the conceptual distinction; C and D state false absolutes. **বাংলা:** মূল পার্থক্য — ফাইন-টিউনিং ওজন বদলায় (gradient update), প্রম্পট/RAG ওজন ছোঁয় না

Q2. A team fine-tunes a base model and then deploys it. They observe that the *same* short prompt now produces noticeably more domain-appropriate answers than before. Which statement best explains this?

- A. A retrieval index was attached automatically during fine-tuning.
- B. The prompt is now being silently expanded with hidden instructions at inference time.
- C. Fine-tuning changed what the model tends to do by default, so the behavior is baked into the weights rather than carried by the prompt.
- D. The model became larger during fine-tuning, increasing its capacity.

> **Answer: C.** Fine-tuning changes the model's default style, format, and domain behavior, so instructions no longer need to ride along in every prompt. A and B invent mechanisms not in the chapter; D is false — fine-tuning does not add parameters to the base. **বাংলা:** ফাইন-টিউনিং ডিফল্ট আচরণ ওজনে গেঁথে দেয়, তাই ছোট প্রম্পটেও ডোমেইন-উপযোগী উত্তর আসে

Q3. Using the chapter's "where does the change live" mental model, which row is correctly matched?

- A. Prompting → in the weights → permanent.
- B. Retrieval-augmented generation → in the context window → per request.
- C. Fine-tuning → in an external index → per request.
- D. Fine-tuning → in the weights → permanent.

> **Answer: D.** The lever table places fine-tuning's change in the weights with permanent persistence. Prompting lives in the context window (per request) and retrieval in an external index (per request), so A, B, and C are mismatched. **বাংলা:** ফাইন-টিউনিংয়ের পরিবর্তন থাকে ওজনে এবং তা স্থায়ী — এটাই সঠিক জোড়া

Q4. Why does the chapter describe the base model as a "generalist" and fine-tuning as giving it a "default specialty"?

- A. Because fine-tuning shifts the model's default behavior toward a domain while it remains the same underlying generalist model.
- B. Because fine-tuning deletes the base model's general knowledge entirely and replaces it with specialist knowledge.
- C. Because the base model cannot answer any question until it has been fine-tuned.
- D. Because a specialist model is always smaller than a generalist model.

Answer: A. Fine-tuning biases the default behavior toward a specialty without rebuilding the model from scratch. B overstates the effect (that would be extreme catastrophic forgetting, not the intended outcome); C and D are false. **বাংলা:** বেস মডেল সাধারণজ্ঞ; ফাইন-টিউনিং তার ডিফল্ট ঝোঁক একটি ডোমেইনের দিকে সরায়, মডেল বদলে ফেলে না

Topic: Why and When to Fine-tune

Q5. Which scenario is the *best* fit for fine-tuning rather than prompting or retrieval?

- A. A system that must always format every output as a fixed JSON schema in a consistent company tone.
- B. A legal assistant that must cite the exact source paragraph for every answer.
- C. A chatbot that must answer with today's constantly changing stock prices.
- D. A prototype where the developer wants to try ten different instruction styles in one afternoon.

Answer: A. Persistent style and format adoption is the canonical fine-tuning use case. C needs fresh knowledge (retrieval), B needs provenance (retrieval), and D is a quick experiment (prompting). **বাংলা:** স্থায়ী স্টাইল/ফরম্যাট শেখানো — ঠিক এখানেই ফাইন-টিউনিং সেরা

Q6. A company's knowledge base (prices, news, regulations) changes daily. Why does the chapter say fine-tuning is the wrong tool here?

- A. Because fine-tuning cannot handle numerical data such as prices.
- B. Because a fine-tuned model is frozen at training time and cannot reflect facts that change after training.
- C. Because fine-tuning always requires citations, which prices do not have.
- D. Because daily-changing data is too small a dataset to fine-tune on.

Answer: B. Weights are frozen at training time, so volatile knowledge goes stale; retrieval keeps an updatable index. A is false (numbers are fine), C reverses the citation point, and D misattributes the problem to dataset size. **বাংলা:** ফাইন-টিউনড মডেল ট্রেনিং-সময়ে জমে যায়, তাই প্রতিদিন বদলানো তথ্য ধরতে পারে না — RAG লাগে

Q7. You have only about 40 labelled examples for a new task. What does the chapter recommend, and why?

- A. Fine-tune, because even tiny datasets benefit most from gradient updates.
- B. Use full fine-tuning specifically, since parameter-efficient methods need more data.
- C. Build a retrieval index from the 40 examples and never prompt.
- D. Use few-shot prompting, because with fewer than roughly one hundred examples it is usually stronger and far cheaper.

Answer: D. Below roughly one hundred examples, few-shot prompting typically beats fine-tuning at a fraction of the cost. A and B push fine-tuning where data is too scarce; C misapplies retrieval to what is really a behavior/few-shot situation. **বাংলা:** ~১০০-এর কম উদাহরণে few-shot prompting সাধারণত ভালো ও সস্তা

Q8. Why does the chapter insist on having an evaluation harness *before* fine-tuning?

- A. Because the harness speeds up the training loop.
- B. Because without measuring before and after, fine-tuning can silently degrade other capabilities and you would not notice.
- C. Because the harness is required to compute the LoRA scaling factor.

- D. Because regulators require an evaluation harness by law.

Answer: B. Fine-tuning can quietly damage capabilities; you must be able to measure before/after to catch regressions. A and C invent unrelated roles; D is not claimed in the chapter. **বাংলা:** মাপার ব্যবস্থা না থাকলে ফাইন-টিউনিং চুপচাপ অন্য দক্ষতা নষ্ট করে — তাই আগে eval harness

Q9. A regulated bank must show users the exact source document for each answer and must honor deletion requests. Which statement best captures why retrieval is preferred over fine-tuning here?

- A. Retrieval is always faster than a fine-tuned model at inference.
- B. Fine-tuning cannot represent legal language, but retrieval can.
- C. Retrieval can point at sources and its index is deletable, whereas weights cannot cite sources and cannot easily "forget" a document.
- D. Retrieval requires no up-front cost, whereas fine-tuning does.

Answer: C. Provenance and the right-to-be-forgotten both favor an external, updatable, deletable index; weights store behavior, not attributable, removable facts. A is not a general truth, B is false, and D ignores the index-build cost. **বাংলা:** RAG উৎস দেখাতে পারে ও নথি মুছে ফেলা যায়; ওজন থেকে তা সম্ভব নয়

Q10. According to the chapter's rule of thumb, which mapping is correct?

- A. Fine-tune for knowledge, retrieve for behavior, prompt for safety.
- B. Fine-tune for citations, retrieve for style, prompt for compliance.
- C. Fine-tune for freshness, retrieve for format, prompt for accuracy.
- D. Fine-tune for behavior, retrieve for knowledge, prompt for quick experiments.

Answer: D. The lecture's one-liner is "fine-tune for behavior, retrieve for knowledge, prompt for quick experiments." The other options swap the roles. **বাংলা:** আচরণে ফাইন-টিউন, জ্ঞানে RAG, দ্রুত পরীক্ষায় প্রম্পট

Q11. The decision function `choose_adaptation` returns "Use retrieval-augmented generation" first when `needs_citations or knowledge_freshness_days < 30`. What does this ordering tell you about the chapter's priorities?

- A. Citation and freshness requirements are treated as hard knock-out conditions that override the data-size and harness checks.
- B. Citations and freshness are minor concerns checked only after data size.
- C. Retrieval is the default answer for every input regardless of the arguments.
- D. The function prefers fine-tuning whenever any data exists.

Answer: A. Because that branch is evaluated first, needing citations or fresh knowledge immediately routes to retrieval before any data-size or harness logic runs. B understates it, and C and D misread the control flow. **বাংলা:** citation বা তাজা-তথ্যের শর্ত আগে যাচাই হয় — এগুলো অন্য সব শর্তকে ছাপিয়ে যায়

Q12. Which of the following is a failure mode of *prompting alone* that the chapter says motivates fine-tuning?

- A. The index becomes too large to search quickly.
- B. The optimizer states no longer fit on a single graphics card.
- C. The quantization error exceeds half a step.
- D. Long prompts cost tokens on every request and few-shot examples crowd out the actual task input.

Answer: D. Prompting's weaknesses include per-request token cost, ignored instructions under context pressure, inconsistent style, and crowded context. A is a retrieval concern; B and C belong to training/quantization, not prompting. **বাংলা:** লম্বা প্রম্পটে প্রতিবার টোকেন খরচ আর few-shot উদাহরণে আসল ইনপুট চাপা পড়ে — তাই ফাইন-টিউনিং

Q13. A team can already get acceptable accuracy with careful prompting but the prompts have stopped improving despite more effort, and they have a few thousand labelled examples. What does the chapter suggest?

- A. Fine-tuning is now appropriate, since prompting quality has plateaued and labelled examples exist.
- B. Switch to retrieval because prompting failed.
- C. Keep adding more few-shot examples indefinitely.
- D. Reduce the dataset to under one hundred examples first.

Answer: A. Fine-tuning gives a task-specific accuracy boost precisely when prompting has plateaued and labelled data is available. C ignores diminishing returns, B misdiagnoses the situation, and D throws away useful data. **বাংলা:** প্রম্পটিং থমকে গেছে আর লেবেলড ডেটা আছে — এটাই ফাইন-টিউনিংয়ের সময়

Topic: Full Fine-tuning — Definition and Loss

Q14. Which statement best describes what full fine-tuning optimizes?

- A. It updates only a small low-rank subset of parameters to minimize the task loss.
- B. It updates only the quantization scales of the weights.
- C. It updates no parameters and instead changes the prompt.
- D. It updates every parameter of the base model to minimize the task (negative log-likelihood) loss.

Answer: D. Full fine-tuning updates all parameters θ by minimizing the negative log-likelihood on the fine-tuning data. A is parameter-efficient fine-tuning, B is quantization, and C is prompting. **বাংলা:** Full fine-tuning সব প্যারামিটার আপডেট করে task loss কমাতে

Q15. In the regularized objective `L_total = L_task +` $(\lambda/2)\cdot||\theta - \theta$`_pretrained||`^{2}, what happens as λ becomes very large?

- A. The model is pulled hard toward the pretrained weights, reducing forgetting but limiting how much new behavior it can learn.
- B. The model can move arbitrarily far from the pretrained weights, learning the new task perfectly.
- C. The task loss is ignored entirely and only quantization matters.
- D. The learning rate automatically increases to compensate.

Answer: A. A large λ taxes deviation from θ_pretrained, so forgetting drops but adaptation capacity shrinks. B describes small λ, C misreads the objective, and D invents a coupling not present. **বাংলা:** λ বড় হলে মডেল প্রি-ট্রেইনড অবস্থানের কাছে আটকে থাকে — ভুলে যাওয়া কমে, কিন্তু নতুন শেখাও কমে

Q16. The penalty term treats θ_pretrained as a "prior." Which statement best describes the intuition?

- A. Deviating from the pretrained weights is encouraged because the prior is uninformative.
- B. The prior forces all weights to zero.
- C. Deviating from the pretrained weights is penalized quadratically, so the model prefers solutions close to where it started unless the task loss strongly pulls it away.
- D. The prior replaces the task loss completely.

Answer: C. The quadratic penalty taxes deviation from the pretrained point, biasing the solution to stay near it. A reverses the sign of the effect, and B and D misstate what the penalty does. **বাংলা:** প্রি-ট্রেইনড বিন্দু থেকে সরে যাওয়া quadratically শাস্তি পায় — মডেল কাছেই থাকতে চায়

Q17. Elastic weight consolidation refines the penalty to $(\lambda/2)\cdot\Sigma$ `F_i` $(\theta$`_i` $- \theta$`_pretrained,i`$)^2$. What is the role of the importance weights F_i?

- A. They scale the learning rate per layer.
- B. They set the rank of the LoRA update.
- C. They convert the weights to 4-bit precision.
- D. They weight each coordinate by its estimated importance for old capabilities, so important weights are protected more strongly from change.

Answer: D. F_i estimates how important each coordinate is for previously learned capabilities, so high-F_i weights resist change more. A, B, and C describe unrelated mechanisms. **বাংলা:** F_i প্রতিটি ওজনের পুরোনো-দক্ষতার গুরুত্ব মাপে — গুরুত্বপূর্ণ ওজনকে বেশি রক্ষা করে

Topic: Catastrophic Forgetting

Q18. Which statement best describes the *mechanism* of catastrophic forgetting in full fine-tuning?

- A. The model runs out of memory and drops random weights.
- B. Gradient descent moves all weights to reduce loss on a narrow slice of data, overwriting weights that encoded unrelated capabilities because nothing in the new loss protects them.
- C. The quantization step rounds important weights to zero.
- D. The retrieval index overwrites the model's knowledge.

Answer: B. With all weights free and a narrow loss, the optimizer overwrites unprotected weights that held other capabilities. A and C describe memory/quantization artifacts, not forgetting; D involves retrieval, which is not part of this mechanism. **বাংলা:** সরু ডেটায় গ্রেডিয়েন্ট সব ওজন নাড়ে; পুরোনো দক্ষতার ওজন রক্ষার শর্ত না থাকায় তা মুছে যায়

Q19. A model is fully fine-tuned on a narrow medical-coding dataset. What does the chapter predict will happen to its general reasoning and multilingual abilities?

- A. They will measurably degrade, even though the medical-coding performance improves.
- B. They will be unaffected, because only medical weights change.
- C. They will improve, because all training improves all skills.
- D. They will be deleted only if quantization is used.

Answer: A. The consequence of catastrophic forgetting is that general capabilities degrade measurably while the narrow task improves. C and B deny the documented trade-off; D ties it to an unrelated technique. **বাংলা:** সরু ডেটায় full fine-tuning করলে টাস্ক ভালো হয় কিন্তু সাধারণ দক্ষতা মাপযোগ্যভাবে কমে

Q20. Why does the chapter call parameter-efficient fine-tuning a way to take the regularization idea "to the limit"?

- A. Because it uses an infinitely large λ on every weight.
- B. Because it removes the task loss entirely.
- C. Because instead of merely taxing deviation, it structurally constrains it — the base stays exactly frozen and updates are confined to a small subspace.
- D. Because it trains for an unlimited number of steps.

Answer: C. PEFT structurally constrains deviation (frozen base, rank-r subspace) rather than softly penalizing it. A is a loose analogy but not the chapter's framing, and B and D are false. **বাংলা:** PEFT শুধু শাস্তি দেয় না, কাঠামোগতভাবে সীমা টানে — বেস জমাট, পরিবর্তন ছোট সাবস্পেসে

Q21. Which mitigation for catastrophic forgetting is correctly described?

- A. Data mixing with pretraining-like data, a small learning rate, and early stopping during full fine-tuning.
- B. Increasing the learning rate so the model adapts faster.
- C. Removing the evaluation harness to avoid overfitting to it.
- D. Quantizing the base to 4 bits, which prevents any forgetting.

Answer: A. The chapter lists data mixing, a small learning rate, and early stopping as a forgetting-mitigation recipe for full fine-tuning. B would worsen drift, C is reckless, and D addresses memory, not forgetting. **বাংলা:** প্রি-ট্রেইনিং-সদৃশ ডেটা মেশানো, ছোট learning rate, early stopping — এগুলোই forgetting কমায়

Q22. A key advantage of freezing the base in Low-Rank Adaptation, relative to forgetting, is that:

- A. The adapter can never be removed once added.
- B. The base weights are retrained alongside the adapter for safety.
- C. Removing (or unmerging) the adapter restores the original model exactly, so the base capabilities are structurally protected.
- D. The model forgets the new task instead of the old one.

Answer: C. Because the base stays exactly frozen, unmerging the factors recovers the original model precisely, bounding any capability loss. A is the opposite of the truth, B contradicts "frozen," and D is nonsense. **বাংলা:** বেস জমাট বলে অ্যাডাপ্টার খুলে ফেললে আদি মডেল হুবহু ফিরে আসে — পুরোনো দক্ষতা সুরক্ষিত

Topic: Training-Memory Arithmetic

Q23. Mixed-precision training with the Adam optimizer keeps about 16 bytes per parameter. Which breakdown is correct?

- A. 4 (weights) + 4 (gradients) + 4 (master copy) + 4 (one Adam moment).
- B. 8 (weights) + 8 (gradients), with no optimizer states.
- C. 2 (weights) + 2 (gradients) + 2 (master copy) + 2 (Adam moment).
- D. 2 (16-bit weights) + 2 (16-bit gradients) + 4 (32-bit master copy) + 4 (Adam first moment) + 4 (Adam second moment).

Answer: D. The chapter's accounting is 2 + 2 + 4 + 4 + 4 = 16 bytes. A omits a moment and mis-sizes the weights/gradients, B ignores optimizer state, and C undercounts the 32-bit components. **বাংলা:** ২+২+৪+৪+৪ = ১৬ বাইট — ১৬-বিট ওজন ও গ্রেডিয়েন্ট, ৩২-বিট মাস্টার কপি ও দুই Adam মোমেন্ট

Q24. Why does the master copy of the weights and the Adam moments each take 4 bytes while the working weights take only 2?

- A. The master copy and moments are stored in 32-bit float for numerical stability, whereas the working weights and gradients are 16-bit.
- B. The master copy is stored in 4-bit and padded to 4 bytes.
- C. The working weights are stored in 8-bit, halving their size.
- D. The moments are duplicated across two graphics cards.

Answer: A. Mixed precision keeps a 32-bit (4-byte) master copy and 32-bit optimizer moments for stability, with 16-bit (2-byte) working weights and gradients. B, C, and D misstate the precisions. **বাংলা:** মাস্টার কপি ও মোমেন্ট ৩২-বিটে (স্থিতিশীলতার জন্য), কাজের ওজন/গ্রেডিয়েন্ট ১৬-বিটে

Q25. A 7B-parameter model is fully fine-tuned with mixed-precision Adam. Roughly how much memory do the parameter states require (excluding activations)?

- A. About 14 GB.
- B. About 28 GB.
- C. About 112 GB.
- D. About 3.63 GB.

Answer: C. $7 \times 10^9 \times 16$ bytes $= 112 \times 10^9$ bytes $= 112$ GB. A is the 16-bit weights alone, B doubles that, and D is the QLoRA figure. **বাংলা:** $৭\times১০^9 \times ১৬$ বাইট = ১১২ GB

Q26. Why does the chapter say full fine-tuning of a 7B model cannot run on a single commodity graphics card?

- A. Because the model has too many layers to load.
- B. Because the 112 GB of parameter states exceeds the 24–80 GB of a single commodity card, so multi-graphics-card sharding is required.
- C. Because commodity cards cannot run 16-bit arithmetic.
- D. Because the optimizer is incompatible with single-card training.

Answer: B. 112 GB far exceeds any single 24–80 GB card, forcing sharding across multiple cards. A misattributes the cause to layer count, and C and D are false hardware claims. **বাংলা:** ১১২ GB একটি ২৪–৮০ GB কার্ডে আঁটে না, তাই একাধিক কার্ডে ভাগ করতে হয়

Q27. In the comparison table, Low-Rank Adaptation at r=8 on a 7B model needs about 14.13 GB while QLoRA needs about 3.63 GB. What is the dominant reason QLoRA is so much smaller?

- A. QLoRA trains far fewer adapter parameters than Low-Rank Adaptation.
- B. QLoRA removes the optimizer states entirely.
- C. QLoRA stores the frozen base in 4-bit (0.5 bytes per parameter) instead of 16-bit (2 bytes), shrinking the dominant base-weight cost roughly fourfold.
- D. QLoRA uses a smaller model than Low-Rank Adaptation.

Answer: C. Both keep tiny ~0.13 GB adapters; the base drops from 14 GB (16-bit) to 3.5 GB (4-bit), which is the ~4× saving. A is false (adapter counts are equal), and B and D are false. **বাংলা:** বেসকে ৪-বিটে রাখায় (০.৫ বাইট/প্যারামিটার) মূল ওজনের খরচ ~৪ গুণ কমে — অ্যাডাপ্টার দুটিতেই ছোট

Q28. For Low-Rank Adaptation at r=8 on a 7B model, the base weights occupy about 14 GB but the trainable adapter states only about 0.13 GB. What does this contrast illustrate?

- A. That the adapters are stored in 4-bit precision.
- B. That Low-Rank Adaptation trains the base weights as well as the adapters.
- C. That activations dominate the memory in all cases.
- D. That gradients and optimizer states are needed only for the tiny adapter, while the large frozen base needs just its forward weights — which is why parameter-efficient fine-tuning saves so much memory.

Answer: D. The expensive 16-byte-per-parameter accounting applies only to the small trainable adapter; the frozen base needs just 2 bytes per weight for the forward pass. A misstates adapter precision, B contradicts "frozen," and C is not the point here. **বাংলা:** গ্রেডিয়েন্ট ও অপটিমাইজার-স্টেট লাগে শুধু ছোট অ্যাডাপ্টারে; জমাট বেসের শুধু forward-ওজন লাগে — এতেই বিশাল সাশ্রয়

Topic: Matrix Rank

Q29. Which statement best defines the rank of a matrix as used in this chapter?

- A. The number of linearly independent rows (equivalently columns) of the matrix.
- B. The total number of entries in the matrix.
- C. The largest entry of the matrix.
- D. The number of layers the matrix appears in.

Answer: A. Rank is the number of linearly independent rows (equivalently columns). B is the entry count, and C and D are unrelated. **বাংলা:** Rank = ম্যাট্রিক্সের স্বাধীন সারি (বা কলাম)-সংখ্যা

Q30. For the 2×2 example $\Delta W = uv^T$ with $u = (2,1)^T$ and $v = (3,-1)^T$, the second row equals 0.50 times the first row and the determinant is 0. What is the rank, and why?

- A. Rank 2, because it is a 2×2 matrix.
- B. Rank 1, because the rows are linearly dependent, so only one independent direction exists.
- C. Rank 0, because the determinant is zero.
- D. Rank 4, because there are four entries.

Answer: B. An outer product uv^T has rank 1; the dependent rows and zero determinant confirm one independent direction. A confuses size with rank, C is only true for the zero matrix, and D confuses entries with rank. **বাংলা:** outer product মানে rank 1 — সারিগুলো পরস্পর-নির্ভর, স্বাধীন দিক মাত্র একটি

Q31. For a square matrix with $d = k = 4096$ and rank $r = 8$, the low-rank factors store $r(d+k) = 65{,}536$ numbers versus $dk = 16{,}777{,}216$ for the full matrix. What general lesson does the chapter draw?

- A. Low-rank storage saves nothing at any size.
- B. The full matrix is always cheaper to store than its factors.
- C. The storage saving grows with matrix size, which is exactly why low-rank updates matter for large language models.
- D. The saving only appears for tiny 2×2 matrices.

Answer: C. At toy 2×2 size there is no saving, but for large d and k the factors are far smaller, so savings scale up — the motivation for LoRA. A and B contradict the numbers, and D reverses the toy-versus-large lesson. **বাংলা:** ম্যাট্রিক্স যত বড়, low-rank সঞ্চয় তত বেশি — তাই বড় মডেলে এটা কাজে আসে

Q32. Why is the *fine-tuning update* ΔW believed to be approximately low-rank?

- A. Because the update is always exactly zero.
- B. Because all weight matrices are symmetric.
- C. Because quantization forces the update into low rank.
- D. Because the pretrained model already contains the needed features, so adaptation mostly re-combines existing directions, giving ΔW a fast-decaying singular-value spectrum.

Answer: D. Adaptation re-weights existing features rather than learning new ones, so the ideal ΔW concentrates in a few directions (low intrinsic dimension, fast-decaying singular values). A is false,

and B and C are unrelated. **বাংলা:** দরকারি ফিচার আগেই শেখা; নতুন কাজ শুধু পুরোনো দিকগুলোর নতুন মিশ্রণ চায় — তাই ΔW প্রায় low-rank

Q33. The chapter cites the *intrinsic dimension* (Aghajanyan et al. 2020). Which statement best describes it?

- A. The number of bits used to store each weight.
- B. The smallest subspace dimension in which a task can be learned well — often hundreds of directions, not millions.
- C. The number of transformer layers in the model.
- D. The number of training examples required.

Answer: B. Intrinsic dimension is the smallest subspace dimension in which the task can be learned well, and it is surprisingly small. A, C, and D describe precision, depth, and dataset size respectively. **বাংলা:** Intrinsic dimension = কাজ ভালোভাবে শেখার ন্যূনতম সাবস্পেস-মাত্রা — প্রায়ই কয়েকশো, লক্ষ নয়

Q34. A rank-r truncation of ΔW is "the best possible" by the Eckart–Young theorem. What does this guarantee in plain terms?

- A. That any rank-r approximation works equally well.
- B. That the truncation always equals the full matrix.
- C. That keeping the top-r singular directions gives the best achievable rank-r approximation, so when the spectrum decays fast, little quality is lost.
- D. That truncation increases the rank beyond r.

Answer: C. Eckart–Young says the top-r singular components form the optimal rank-r approximation; with a fast-decaying spectrum, the loss is small. A, B, and D contradict the theorem. **বাংলা:** শীর্ষ-r সিঙ্গুলার দিক রাখা মানেই সেরা rank-r আনুমানিক — স্পেকট্রাম দ্রুত পড়লে মানের ক্ষতি সামান্য

Topic: Low-Rank Adaptation (LoRA) — Formulation

Q35. Which statement best describes the Low-Rank Adaptation update applied to a frozen weight matrix W?

- A. W stays frozen and a trainable low-rank term $(\alpha/r)\cdot B\cdot A$ is added to it.
- B. W is replaced entirely by a new trainable matrix of the same size.
- C. W is quantized to 4 bits and then retrained.
- D. W is multiplied by a trainable scalar.

Answer: A. LoRA forms W' = W + (α/r)BA with W frozen and only B, A trainable. B is full fine-tuning, C is quantization, and D is far too restrictive. **বাংলা:** W জমাট থাকে; তাতে যোগ হয় শেখানো low-rank পদ (α/r)BA

Q36. In W' = W + (α/r)BA with A $\in \mathbb{R}^{r\times k}$ and B $\in \mathbb{R}^{d\times r}$, why is the update BA constrained to rank at most r?

- A. Because A and B are square matrices.
- B. Because a product of a d×r matrix and an r×k matrix can have rank at most r, the smaller inner dimension.
- C. Because α divides the update.
- D. Because W is frozen.

Answer: B. The product BA passes through the bottleneck of width r, so its rank cannot exceed r. A is false (they are not square), C confuses scaling with rank, and D concerns W, not the rank of BA. **বাংলা:** d×r ও r×k গুণফলের rank সর্বোচ্চ r (ছোট ভেতরের মাত্রা)

Q37. Low-Rank Adaptation initializes A ~ N(0, σ^2) but B = 0. Why is B set to zero?

- A. So the rank of the update becomes zero permanently.
- B. So the base weights are also reset to zero.
- C. So the update BA = 0 at the start, meaning training begins exactly at the pretrained model with no random disturbance of base behavior.
- D. So no gradients ever flow into B.

Answer: C. With B = 0, $\Delta W = BA = 0$ initially, so the model starts identical to the pretrained base; B then learns via gradients. A is false (rank grows during training), B and D contradict the design. **বাংলা:** B=0 দিলে শুরুতে $\Delta W=0$ — মডেল হুবহু প্রি-ট্রেইনড, পরে B শেখে

Q38. Why does the chapter say r and α are *not* redundant hyperparameters?

- A. Because r sets the learning rate and α sets the batch size.
- B. Because α is an integer and r is a float.
- C. Because only one of them is ever used at a time.
- D. Because r controls capacity (how many directions can change) while α controls the effective step size of the update; the α/r scaling decouples magnitude from rank.

Answer: D. r is the capacity knob and α is the effective step-size knob; the α/r factor keeps overall scale comparable as r changes. A, B, and C misstate their roles. **বাংলা:** r ঠিক করে ক্ষমতা (কয়টা দিক বদলাবে), α ঠিক করে ধাপের আকার — α/r স্কেলিং দুটোকে আলাদা রাখে

Q39. If you double the rank r while keeping α fixed, what does the α/r scaling do to the per-direction contribution of the update?

- A. It doubles the per-direction contribution.
- B. It halves the per-direction contribution, keeping the overall update scale comparable.
- C. It leaves the contribution unchanged.
- D. It sets the contribution to zero.

Answer: B. Since the update is scaled by α/r, doubling r halves the scaling factor, so each direction contributes about half as much, stabilizing the overall scale. A reverses it, and C and D are wrong. **বাংলা:** r দ্বিগুণ করলে α/r অর্ধেক — প্রতি-দিকের অবদান অর্ধেক হয়, সামগ্রিক মাত্রা প্রায় একই থাকে

Q40. During the Low-Rank Adaptation forward pass, the update is computed as $(\alpha/r)\cdot B(Ax)$ rather than first forming the full matrix BA. Why?

- A. Because forming BA would change the result numerically.
- B. Because B must be applied before A.
- C. Because the scaling factor cannot be applied to a full matrix.
- D. Because computing Ax (a small vector) then $B(\cdot)$ uses two skinny matrix multiplications and avoids ever materializing the large $d\times k$ matrix BA, which is far cheaper.

Answer: D. Two skinny multiplications (Ax then $B(\cdot)$) avoid building the large BA matrix, saving compute and memory. A is false (algebraically identical), and B and C are wrong. **বাংলা:** আগে Ax (সরু), পরে $B(\cdot)$ — বড় BA ম্যাট্রিক্স কখনো বানাতে হয় না, তাই সস্তা

Q41. A Low-Rank-Adaptation-adapted layer is *merged* for deployment via $W' = W + (\alpha/r)BA$. What is the consequence at inference time?

- A. Inference becomes slower because of the extra matrix.
- B. Inference is a single dense multiplication of the same shape as the base, so there is zero added latency.
- C. The base model can no longer be recovered under any circumstances.
- D. The adapter parameters double in size.

Answer: B. After merging, the layer is one dense matrix of the original shape, giving zero added latency; it can also be unmerged to restore the base. A is the unmerged case, C is false (unmerging works), and D is invented. **বাংলা:** মার্জ করলে একটিই dense গুণ — base-এর মতোই, latency শূন্য; চাইলে আবার unmerge করা যায়

Q42. Which is listed as a *con* of Low-Rank Adaptation in the chapter?

- A. On very large datasets or strong domain shifts it can fall slightly below full fine-tuning quality, and it adds two hyperparameters (r, α) plus the choice of target matrices.
- B. It cannot be merged for inference.
- C. It always forgets the base task completely.
- D. It requires storing the base model in 4-bit.

Answer: A. LoRA's downsides are a slight quality gap under heavy data/shift and extra hyperparameters/target choices. C contradicts its forgetting-resistance, B is false (it merges), and D describes QLoRA, not LoRA. **বাংলা:** বিশাল ডেটা/বড় ডোমেইন-শিফটে মান একটু কম, আর দুই বাড়তি হাইপারপ্যারামিটার ও টার্গেট-ম্যাট্রিক্স পছন্দ — এগুলোই খুঁত

Q43. A SaaS company wants one shared base model but customer-specific behavior for thousands of tenants. Why is Low-Rank Adaptation a strong fit?

- A. Because each tenant needs a full copy of the base weights.
- B. Because Low-Rank Adaptation adapters are megabyte-sized and hot-swappable, so one base can serve many per-tenant adapters loaded on demand.
- C. Because Low-Rank Adaptation changes the base weights for all tenants at once.
- D. Because Low-Rank Adaptation requires retraining the base per tenant.

Answer: B. Tiny, hot-swappable adapters on a shared frozen base give per-tenant isolation cheaply. A wastes memory, and C and D contradict the frozen-base, per-adapter design. **বাংলা:** ছোট (মেগাবাইট) hot-swappable অ্যাডাপ্টার — এক বেসে বহু টেন্যান্ট, চাহিদামতো লাগানো-খোলা

Topic: Counting LoRA Parameters

Q44. Which formula gives the trainable-parameter count for a *single* Low-Rank-Adaptation-adapted matrix of shape d×k at rank r?

- A. $r \cdot d \cdot k$
- B. $r(d + k)$
- C. $d \cdot k$
- D. $(d + k) / r$

Answer: B. Each LoRA matrix contributes A (r×k) plus B (d×r), i.e., r(d+k) trainable parameters. A is a classic trap (the common mistake), C is the full-matrix count, and D is dimensionally wrong. **বাংলা:** এক ম্যাট্রিক্সে trainable = r(d+k); r·d·k লেখা ক্লাসিক ভুল

Q45. For a 4096×4096 matrix at r = 8, the reduction factor versus full training is 16,777,216 / 65,536 = 256×. What does the general expression dk / r(d+k) tell you about *which* matrices save the most at fixed r?

- A. Smaller, thinner matrices save the most.
- B. The reduction is independent of matrix shape.
- C. Only non-square matrices save anything.
- D. Bigger and squarer matrices give bigger savings at fixed rank.

Answer: D. Since the reduction is dk / r(d+k), it grows for large and square ($d \approx k$) matrices. A reverses the trend, C is false, and B contradicts the formula. **বাংলা:** dk / r(d+k) বড় ও বর্গাকার ম্যাট্রিক্সে বেশি — তাই সেগুলোতেই সঞ্চয় সবচেয়ে বেশি

Q46. Attaching Low-Rank Adaptation at r = 8 to the four attention projections (each 4096×4096) across all 32 layers of a 7B model gives 8,388,608 trainable parameters. What share of the 7-billion base is this?

- A. About 1.2 %.
- B. About 12 %.
- C. About 0.012 %.
- D. About 0.12 %.

Answer: D. 8,388,608 / 7,000,000,000 × 100 ≈ 0.12 %. A is ten times too large, B is a hundred times too large, and C is ten times too small. **বাংলা:** ৮৩,৮৮,৬০৮ / ৭০০ কোটি ≈ ০.১২%

Q47. The general formula for total Low-Rank Adaptation parameters is $N = L \cdot \Sigma\_\{t \in T\}\ r(d\_t + k\_t)$. What does each part contribute conceptually?

- A. L is the rank, T is the learning rate, and r is the layer count.
- B. The formula counts only the base parameters, not the trainable ones.

- C. L is the number of layers, the sum runs over the set of target matrices T (each with its own shape), and r(d_t + k_t) is the per-matrix count.
- D. r appears squared because there are two factors A and B.

Answer: C. You multiply the per-matrix count r(d_t + k_t), summed over targets, by the number of layers L. A scrambles the symbols, B misreads what is counted, and D is false (r is not squared). **বাংলা:** L = লেয়ার সংখ্যা, যোগফল চলে টার্গেট-ম্যাট্রিক্সগুলোর ওপর, r(d_t+k_t) = প্রতি-ম্যাট্রিক্স গণনা

Topic: Adapters (Bottleneck Modules)

Q48. Which statement best describes an adapter module of the form y = x + W_up·GELU(W_down·x)?

- A. A parallel additive low-rank update to an existing weight matrix.
- B. A 4-bit quantization of the base weights.
- C. A small residual bottleneck multilayer perceptron inserted as a new sequential module after a transformer sub-layer, with a nonlinearity in the middle.
- D. A scaling factor applied to the attention scores.

Answer: C. Adapters are inserted bottleneck modules (down-project, GELU, up-project) with a residual connection. A describes Low-Rank Adaptation, B is quantization, and D is unrelated. **বাংলা:** অ্যাডাপ্টার = লেয়ারের পরে বসানো ছোট bottleneck MLP, মাঝে GELU, residual সহ

Q49. Why can a Low-Rank Adaptation update be merged into the base weight while an adapter module cannot?

- A. Because the adapter has fewer parameters than Low-Rank Adaptation.
- B. Because the Low-Rank Adaptation update is linear in x and so folds into one matrix, whereas the adapter's GELU nonlinearity cannot be absorbed into any neighboring weight matrix.
- C. Because adapters are stored in 4-bit precision.
- D. Because Low-Rank Adaptation uses a residual connection and adapters do not.

Answer: B. Linearity lets Wx + ΔWx collapse to (W + ΔW)x; the adapter's GELU breaks this, forcing sequential computation. A is irrelevant to mergeability, C is false, and D reverses the residual fact. **বাংলা:** LoRA রৈখিক বলে এক ম্যাট্রিক্সে মেশে; অ্যাডাপ্টারের GELU (nonlinearity) মেশানো আটকায়

Q50. What is the practical inference-time consequence of the adapter's non-mergeability?

- A. The adapter adds an extra sequential computation per layer on every request, permanently increasing inference latency.
- B. The adapter makes inference faster than the base model.
- C. The adapter has no effect on inference at all.
- D. The adapter must be retrained on every request.

Answer: A. Because it cannot be folded in, the adapter's bottleneck runs sequentially every layer, every request, adding latency. B and C contradict that, and D is nonsense. **বাংলা:** মেশানো যায় না বলে প্রতি লেয়ারে প্রতি অনুরোধে বাড়তি গণনা চলে — latency বাড়ে

Q51. Comparing the two methods at d = 4096, r = 8, L = 32, the chapter reports adapters at ≈ 4.46 M parameters and Low-Rank Adaptation (q,k,v,o) at ≈ 8.39 M. Which statement is the *most accurate* takeaway?

- A. Adapters always have more trainable parameters than Low-Rank Adaptation.
- B. Low-Rank Adaptation and adapters are identical in every respect.
- C. In this specific configuration adapters happen to have fewer trainable parameters, but they add inference latency, whereas Low-Rank Adaptation can be merged for zero added latency.
- D. Adapters are mergeable and Low-Rank Adaptation is not.

Answer: C. The parameter counts depend on the configuration (here adapters are fewer), but the decisive practical difference is latency: LoRA merges to zero, adapters do not. A overgeneralizes, B is false, and D reverses the mergeability fact. **বাংলা:** এই কনফিগে অ্যাডাপ্টারে প্যারামিটার কম, কিন্তু latency বাড়ায়; LoRA মার্জ করে latency শূন্য — এটাই আসল পার্থক্য

Topic: Quantization and Affine Quantization

Q52. Which statement best describes affine (asymmetric) quantization?

- A. It stores weights at higher precision to reduce error.
- B. It maps a real interval [x_min, x_max] onto integers {0,…,2^b−1} using a scale s and a zero-point z.
- C. It removes the least important weights from the matrix.
- D. It increases the rank of the weight matrix.

Answer: B. Affine quantization maps the real range onto b-bit integers via scale and zero-point. A reverses the goal (it lowers precision), and C and D describe pruning and rank, not quantization. **বাংলা:** Affine quantization বাস্তব রেঞ্জকে s ও z দিয়ে b-বিট পূর্ণসংখ্যায় ম্যাপ করে

Q53. In affine quantization the maximum reconstruction error is bounded by s/2. What does a *larger* scale s imply?

- A. Smaller steps and therefore smaller maximum error.
- B. That the zero-point z must be negative.
- C. That the bit width b has increased.
- D. Coarser steps and therefore a larger maximum rounding error.

Answer: D. s is the real width of one integer step; a larger s means coarser quantization and a larger worst-case error ($\leq$ s/2). A reverses it, and B and C are unrelated. **বাংলা:** s বড় মানে ধাপ মোটা — সর্বোচ্চ ভুল ($\leq$ s/2) বেশি

Q54. For 4-bit weights over the range [−2.00, 1.00], the scale is s = 3.00/15 = 0.20. If the range were instead [−1.00, 1.00] at the same 4 bits, what happens to s and the worst-case error?

- A. Both increase.
- B. Both decrease, because a narrower range divided into the same 16 levels gives finer steps (s = 2.00/15 $\approx$ 0.13) and a smaller s/2 error.
- C. s decreases but the error increases.
- D. Neither changes, because the bit width is the same.

Answer: B. With 16 levels fixed, a narrower range yields a smaller step s ($\approx$ 0.13) and thus a smaller s/2 bound. A and C get the direction wrong, and D ignores that s depends on the range, not only b. **বাংলা:** রেঞ্জ সরু হলে একই ১৬ ধাপে s ছোট হয় ($\approx$০.১৩), তাই সর্বোচ্চ ভুলও কমে

Q55. Storage per parameter is 2 bytes at 16-bit, 1 byte at 8-bit, and 0.5 bytes at 4-bit. Why does this progression matter for QLoRA's headline result of fine-tuning a 65B model on a single 48 GB card?

- A. Because 0.5 bytes per parameter puts the 65B *frozen base* at about 32.50 GB, leaving room on a 48 GB card for adapters and activations.
- B. Because 4-bit storage removes the need for any Low-Rank Adaptation factors.
- C. Because 8-bit storage is what enables the 65B result.
- D. Because the optimizer states shrink to 4 bits as well.

Answer: A. $65 \times 10^9 \times 0.5$ bytes = 32.5 GB for the base fits within 48 GB with headroom for the 16-bit adapters and activations. B and C are false, and D is wrong — the trainable side stays at 16 bytes per parameter. **বাংলা:** ০.৫ বাইট/প্যারামিটারে ৬৫B বেস $\approx$ ৩২.৫ GB — ৪৮ GB কার্ডে অ্যাডাপ্টার ও অ্যাক্টিভেশনের জায়গা থাকে

Topic: QLoRA

Q56. Which statement best describes QLoRA?

- A. The base model is quantized to 4 bits and also trained in 4 bits.
- B. The Low-Rank Adaptation factors are quantized to 4 bits while the base stays in 16 bits.
- C. The frozen base model is stored in 4 bits while 16-bit Low-Rank Adaptation factors are trained on top.
- D. The optimizer states are quantized to 4 bits while everything else stays in 16 bits.

Answer: C. QLoRA freezes and stores the base in 4-bit (NormalFloat4, double quantization) and trains 16-bit LoRA factors on top, with a paged optimizer. A is wrong (the base is never trained in 4-bit), B reverses what is quantized, and D describes a different 4-bit-optimizer technique. **বাংলা:** QLoRA = জমাট বেস ৪-বিটে সংরক্ষিত + ১৬-বিট LoRA ফ্যাক্টর শেখানো; বেস ৪-বিটে ট্রেইন হয় না

Q57. In the QLoRA training step, gradients flow only into A and B, while the base is dequantized on the fly during the forward pass. What trade-off does this create?

- A. It saves both memory and compute with no downside.
- B. It increases memory but reduces compute.
- C. It trains the base weights in 4-bit, risking instability.
- D. It saves memory but adds some extra compute per step to dequantize the 4-bit blocks.

Answer: D. The ~4× memory cut versus a 16-bit base is paid for with extra per-step dequantization compute; gradients still only reach A and B. A ignores the compute cost, B reverses the memory effect, and C contradicts "frozen base." **বাংলা:** মেমরি বাঁচে (~৪×), কিন্তু প্রতি স্টেপে ৪-বিট ব্লক ডিকোয়ান্টাইজ করতে বাড়তি গণনা লাগে; গ্রেডিয়েন্ট শুধু A,B-তে

Q58. Why does QLoRA keep the small Low-Rank Adaptation factors A and B in 16-bit precision rather than also quantizing them to 4-bit?

- A. Because the factors are the only parameters that receive gradients, and training them needs the numerical precision; they are tiny, so keeping them in 16-bit costs almost no memory.
- B. Because 4-bit factors would be faster to train.
- C. Because the factors must match the 4-bit base exactly.
- D. Because 16-bit factors eliminate the need for a paged optimizer.

Answer: A. The trainable factors need adequate precision for stable gradient updates, and since they are a tiny fraction of parameters (~0.13 GB), 16-bit storage is essentially free. B, C, and D are not supported by the chapter. **বাংলা:** A,B-ই কেবল শেখে, তাই precision দরকার; এগুলো ক্ষুদ্র বলে ১৬-বিটে রাখলেও মেমরি প্রায় বাড়ে না

Topic: Decision Framework, Merging, and Task Arithmetic (Synthesis)

Q59. An engineer serves two separately trained Low-Rank Adaptation adapters at once by adding their updates: W' = W + ΔW_legal + ΔW_JSON. According to the chapter, when is such "task arithmetic" most likely to work well?

- A. Always, because Low-Rank Adaptation updates are additive by construction.
- B. When the two updates act in nearly orthogonal subspaces, so their singular directions barely overlap and cross-talk is minimal.
- C. Only when both adapters were trained at 4-bit precision.
- D. Never, because adapters can never be combined.

Answer: B. Summing deltas works best when their subspaces are near-orthogonal (e.g., "legal tone" vs. "JSON structure" are fairly disjoint skills); overlap causes destructive interference. A ignores interference risk, C is irrelevant, and D is too absolute. **বাংলা:** আপডেট দুটির সাবস্পেস প্রায় orthogonal হলে যোগ করলে কম সংঘর্ষ — তখনই task arithmetic ভালো চলে

Q60. What is the central *interference risk* when naively adding two Low-Rank Adaptation updates that share overlapping subspaces?

- A. The base model is permanently destroyed.
- B. The model's parameter count doubles at inference.
- C. Where the subspaces overlap, the deltas combine destructively or inflate in magnitude, distorting both behaviors — and the composition is out-of-distribution since neither adapter was trained with the other present.
- D. The quantization scale becomes negative.

Answer: C. Overlapping directions cause sign conflicts and magnitude inflation, and the combined behavior was never seen in training (out-of-distribution). A overstates it, and B and D are unrelated. **বাংলা:** সাবস্পেস মিলে গেলে আপডেট পরস্পরকে বিকৃত করে; দুই অ্যাডাপ্টার কখনো একসাথে ট্রেইন হয়নি বলে ফল out-of-distribution

Q61. Which is a *safer* alternative to naive adapter addition that the chapter endorses, with its trade-off correctly stated?

- A. Joint training of a single adapter on a mixed dataset — best quality, but requires a new training run and re-evaluation whenever either skill changes.
- B. Increasing the rank of both adapters to 256 so they never interfere.
- C. Deleting the evaluation harness to avoid measuring interference.
- D. Quantizing both adapters to 4-bit so they merge cleanly.

Answer: A. Joint training on a mixed dataset avoids interference at the cost of a fresh training run and re-evaluation; per-request hot-swapping is the other endorsed option. B, C, and D are not safe remedies. **বাংলা:** মিশ্র ডেটায় একটি অ্যাডাপ্টার joint-train করা নিরাপদ — তবে নতুন ট্রেনিং ও পুনঃমূল্যায়ন লাগে

Q62. A team raises the rank from r = 8 to r = 256. Task accuracy barely moves, but held-out general-capability scores drop and memory rises. Using the concept of intrinsic dimension, which explanation is best?

- A. The task's intrinsic dimension exceeded 256, so more capacity was clearly needed.
- B. Raising the rank reduced the parameter count, saving memory.
- C. The scaling factor α/r became larger, increasing each direction's contribution.
- D. The task's intrinsic dimension was already at or below r = 8, so the extra directions only fit noise and let the model drift further from the pretrained weights, causing mild forgetting and overfitting.

Answer: D. With accuracy already saturated, the intrinsic dimension was ≤ 8; extra rank fits noise and enlarges deviation from θ_pretrained, hurting general scores. A contradicts the saturation, B reverses the memory effect, and C gets the α/r direction wrong. **বাংলা:** কাজের intrinsic dimension আগেই r=8-এর নিচে; বাড়তি rank শুধু নয়েজ ধরে ও বেস থেকে বেশি সরায় — তাই সাধারণ স্কোর পড়ে

Q63. Given the rank-versus-quality finding above, what does the chapter recommend?

- A. Always use the largest rank the memory allows.
- B. Abandon Low-Rank Adaptation and switch to full fine-tuning.
- C. Return to a small rank, sweep r $\in \{4, 8, 16\}$ with fixed α, monitor both task and general held-out scores, and prefer the smallest rank on the task-quality plateau.
- D. Remove α from the formula entirely.

Answer: C. Sweep small ranks, watch both task and general scores, and pick the smallest rank on the plateau. A causes drift and overfitting, B is an overreaction, and D breaks the method. **বাংলা:** ছোট rank-এ ফিরে গিয়ে {4,8,16} সুইপ করো, দুই ধরনের স্কোর দেখো, প্লেটোতে সবচেয়ে ছোট rank বেছে নাও

Q64. Which statement best captures how the three levers can be *combined*, per the chapter?

- A. They are mutually exclusive; choosing one forbids the others.
- B. A fine-tuned model can still use retrieval for facts and good prompts — for example, a small style-only adapter plus retrieval for knowledge is an ideal hybrid.
- C. Retrieval must always replace fine-tuning entirely.
- D. Prompting can only be used before any fine-tuning has occurred.

Answer: B. The chapter stresses the levers combine: fine-tune for style, retrieve for facts, and prompt on top. A, C, and D wrongly treat them as exclusive or ordered. **বাংলা:** তিন লিভার একসাথে চলে — স্টাইলের জন্য ছোট অ্যাডাপ্টার + তথ্যের জন্য RAG, এটাই আদর্শ মিশ্রণ

Answer Key

Q#	Answer	Q#	Answer	Q#	Answer	Q#	Answer
Q1	A	Q17	D	Q33	B	Q49	B
Q2	C	Q18	B	Q34	C	Q50	A
Q3	D	Q19	A	Q35	A	Q51	C
Q4	A	Q20	C	Q36	B	Q52	B
Q5	A	Q21	A	Q37	C	Q53	D
Q6	B	Q22	C	Q38	D	Q54	B
Q7	D	Q23	D	Q39	B	Q55	A
Q8	B	Q24	A	Q40	D	Q56	C
Q9	C	Q25	C	Q41	B	Q57	D
Q10	D	Q26	B	Q42	A	Q58	A
Q11	A	Q27	C	Q43	B	Q59	B
Q12	D	Q28	D	Q44	B	Q60	C
Q13	A	Q29	A	Q45	D	Q61	A
Q14	D	Q30	B	Q46	D	Q62	D
Q15	A	Q31	C	Q47	C	Q63	C
Q16	C	Q32	D	Q48	C	Q64	B

Distribution Count

Letter	Count
A	16
B	16
C	16
D	16
Total	**64**