CarbonBench-FSL: Cost-Aware Few-Shot Evaluation in the Foundation Model Era

Liz Lemma   Future Detective

January 20, 2026

Abstract

Few-shot learning (FSL) is increasingly realized via heterogeneous adaptation mechanisms: in-context learning (ICL) in large language/multimodal models, parameter-efficient finetuning (PEFT), and classical meta-learning. Existing FSL benchmarks largely report accuracy alone, obscuring the decisive deployment constraints of 2026: latency, energy, carbon footprint, and communication (federation). Motivated by recent survey observations that (i) evaluation protocols are inconsistent and (ii) Green AI metrics are missing or nonstandard, we introduce CarbonBench-FSL, a benchmark suite and reporting protocol that treats adaptation cost as a first-class citizen. We formalize episodic evaluation with jointly measured utility and cost random variables, define Pareto-frontier and scalarized leaderboard metrics (e.g., hypervolume, area-under-frontier, accuracy-at-budget), and provide an evaluation harness that outputs statistically valid confidence intervals and rankings. We prove tight sample complexity bounds for estimating frontier metrics and for reliably distinguishing methods, and we give hardness results showing that optimal per-task portfolio selection under global budgets is NP-hard. CarbonBench-FSL enables fair comparison across ICL, finetuning/PEFT, meta-learning, and hybrids under domain shift and resource constraints.

Table of Contents

3. 3. Formal Setup and Notation: episodic task distribution, methods, utility and cost random variables, hardware profiles, measurement noise model.

4. 4. Problem Definition: Cost-Aware Few-Shot Evaluation (CAFSE): define outputs (frontiers, scalar metrics, rankings) and desiderata (fairness, reproducibility, statistical validity).

5. 5. CarbonBench-FSL Benchmark Suite Specification: task families, shift regimes (in-domain vs cross-domain), modalities, and mandatory train/test splits; include federation and continual variants as optional tracks (structure only).

6. 6. Measurement Protocol: standardized accounting for tokens, FLOPs, updated parameters, wall-clock, communication rounds, energy (Joules) and CO_2 proxies; calibration of measurement noise and run-to-run variance; reproducibility invariants.

7. 7. Leaderboard Metrics: Pareto frontier definition; scalarizations (accuracy-at-budget, cost-at-accuracy, hypervolume/AUF); tie-breaking and dominance rules; treatment of stochastic methods and prompt sensitivity in ICL.

8. 8. Evaluation Algorithm: pseudocode for sampling episodes, running methods, recording costs, computing estimators, confidence intervals, and frontier metrics; best practices for caching, warmup, and batching without bias.

9. 9. Theory: Tight sample complexity for estimating cost-aware metrics; ranking stability bounds; minimax lower bounds; hardness of portfolio optimization under budgets and approximation options.

10. 10. Experimental Protocol and Reference Baselines (implementation-facing): method list (SimpleShot, ProtoNets, MAML variants, PEFT finetuning baselines, calibrated ICL), required ablations (prompt ordering, domain shift, budget sweeps); what evidence strengthens claims.

11. 11. Discussion: how CarbonBench-FSL changes conclusions in meta-learning vs transfer vs ICL; recommendations for future benchmark extensions (new modalities, privacy, continual).

12. 12. Limitations and Ethics: measurement dependence on hardware and PUE; risks of carbon proxy misuse; responsible reporting guidelines.

1 Introduction

Few-shot learning (FSL) evaluation was historically organized around a single question: given a small labeled support set, which algorithm attains the highest expected query-set accuracy. That framing was adequate when the primary constraint was labeled-data scarcity and when competing methods operated within a narrow computational regime (e.g., a fixed convolutional backbone with episodic fine-tuning or metric learning). In 2026 this premise is no longer stable. The dominant adaptation mechanisms now span in-context learning (ICL) with long-context foundation models, parameter-efficient fine-tuning (PEFT) with low-rank or prefix adapters, full fine-tuning, retrieval-augmented hybrids, and meta-learners that may themselves invoke large pretrained models. These methods exhibit comparable utility on many benchmarks while differing by orders of magnitude in inference-time token consumption, adaptation FLOPs, wall-clock latency, and measured energy. An accuracy-only leaderboard therefore ceases to be a well-posed proxy for deployability, and it is no longer sufficient even as a scientific summary of algorithmic progress.

We proceed from three observations that, taken together, force a cost-aware evaluation protocol. First, the classical separation between "training-time cost" and "test-time cost" has eroded. In ICL, a nontrivial portion of the effective "training signal" is embedded in the prompt, and the prompt is re-sent (and re-computed) per episode. In PEFT and test-time training variants, gradient-based adaptation is explicitly performed at evaluation time. Thus, the resource footprint that matters is not only pretraining, but also the episodic adaptation+inference footprint; the latter is precisely what a few-shot benchmark exercises. Second, the operative bottlenecks in practical deployments increasingly include context length and energy. Tokens are not an incidental implementation detail: they induce both monetary cost (API pricing), latency (attention complexity), and energy (increased memory traffic and compute). Similarly, energy and carbon reporting have become standard in empirical ML practice, and it is methodologically inconsistent to ignore them in a regime where methods trade utility for energy by construction (e.g., longer prompts, more adaptation steps, larger retrieval corpora). Third, algorithmic choices can arbitrarily inflate costs without commensurate utility gains, so any scalar score that depends only on utility admits pathological dominance failures: two methods can be indistinguishable in expected accuracy while being decisively different in energy or token budget, and an accuracy-only ranking cannot detect that distinction.

Recent surveys of FSL and meta-learning diagnose a second, conceptual mismatch between benchmark design and current practice. Classical episodic meta-learning benchmarked a learner that generalizes by acquiring an initialization or metric from many training tasks, contrasting with transfer learning that reuses a pretrained representation and performs simple adaptation. Over time, the empirical gap narrowed, and survey evidence shows that strong pretrained backbones with straightforward adaptation can match or surpass many specialized meta-learners on standard task distributions, especially when training and test classes are not strictly disjoint in a semantic sense. This calls into question what precisely an FSL benchmark is measuring: is it the quality of a meta-learned inductive bias, the quality of a pretrained representation, or merely the ability to exploit scale? Without a joint accounting of utility and adaptation-time cost, we cannot disentangle whether improvements arise from better algorithms or from increased per-episode computation (more steps, larger contexts, more retrieval). A cost-aware protocol is therefore not only a deployment requirement but also a scientific control: it forces comparisons to be made along a tradeoff surface rather than at a single, unconstrained operating point.

The emergence of ICL further destabilizes legacy evaluation conventions. ICL behaves like a meta-learner instantiated at inference time, yet it is governed by qualitatively different knobs: demonstration selection, prompt length, formatting, and calibration procedures. In the few-shot regime, small formatting and selection changes can shift performance materially, and these changes typically scale token counts and attention costs. Moreover, ICL introduces an additional axis of robustness concern: performance may depend on the order and label distribution of demonstrations, and calibration can vary across tasks and domains. An evaluation that reports only mean accuracy implicitly treats these choices as innocuous hyperparameters, whereas in practice they define a family of points on a utility–cost frontier. If we wish to treat ICL as a first-class FSL method (as current practice compels), then our benchmark must expose and measure these knobs under a standardized protocol.

Green AI considerations supply the final necessity. In 2026, it is common to report energy, hardware configuration, and carbon proxies, and it is also common to observe that measurement noise and systems effects (batching, warmup, clock rates, memory constraints) can dominate naïve FLOPs-based estimates. Thus, a benchmark that purports to compare algorithms must specify a measurement protocol and a hardware profile, and must treat costs as random variables subject to estimation error. In the FSL setting, where evaluation is episodic and inherently stochastic, the same statistical discipline applied to accuracy must be applied to costs. Otherwise, cost reporting degenerates into ad hoc anecdotes that cannot support ranking claims. We therefore treat the joint estimation of utility and costs as part of the benchmark definition rather than an optional add-on.

These considerations motivate CarbonBench-FSL as an evaluation problem rather than merely a dataset collection. Our aim is not to replace utility with cost, but to make explicit that any method induces a joint distribution over $(U_m, \mathbf{C}_m)$ on episodes, and that meaningful comparisons require a multi-objective report. Concretely, instead of asking for the maximum accuracy at an unspecified operating point, we ask for the achievable tradeoff between utility and costs under declared knobs, summarized by an estimated Pareto frontier and by budgeted scalar functionals (e.g., accuracy at a token/energy budget, or hypervolume). This shift resolves the ambiguity noted above: a method that improves accuracy only by multiplying adaptation-time computation is not categorically "better"; it occupies a different region of the tradeoff surface.

Finally, we emphasize that the change in evaluation target demands a change in statistical methodology. Episodic benchmarks are commonly run with modest numbers of tasks, and the resulting rankings can be brittle. When costs are added, brittleness can worsen unless one uses paired designs (identical episodes across methods) and reports uncertainty. In our setting, the correct object is an estimate of an expected utility–cost profile with confidence intervals, not a single point estimate. We therefore build the benchmark around reproducible episodic sampling, standardized instrumentation, and finite-sample guarantees for the reported metrics. Under these constraints, an "accuracy-only" evaluation is not merely incomplete; it is formally incapable of identifying Pareto-dominant methods and therefore cannot serve as an adequate summary of progress in few-shot adaptation in the current methodological landscape.

2 Related Work

Few-shot learning has a mature benchmarking tradition, but its dominant abstractions were designed for regimes in which adaptation-time computation is either negligible or implicitly fixed. The early canonical image benchmarks, exemplified by miniImageNet and tieredImageNet **??**, standardized an episodic protocol (e.g., N-way K-shot classification) and focused attention on the expected query accuracy after adaptation on a small support set. Subsequent benchmarks broadened the task distribution and sought to close loopholes associated with narrow domain focus. Meta-Dataset **?** aggregates multiple vision datasets and introduces heterogeneous episode structure (varying ways, shots, and class imbalance), emphasizing that an algorithm should perform across a mixture of task families rather than a single curated dataset. Meta-Album **?** extends coverage to diverse visual domains with a unified interface and highlights that task difficulty varies substantially across domains. Meta-Omnium **?** further expands the scope and stresses distribution shift across modalities and sources. These efforts move the field toward a more realistic $\mathcal{Q}$, but they largely preserve the single-objective evaluation target: rank methods by mean utility, with costs treated, at best, as informal qualifiers.

A parallel line of work seeks *unified* few-shot evaluation across model families and adaptation mechanisms. Benchmarks and libraries such as

Torchmeta, learn2learn, and broader "few-shot as a service" suites provide standardized episode generation and interfaces, enabling comparisons between metric learning, gradient-based adaptation, and meta-learned initializations. More recent unified efforts attempt to place in-context learning, retrieval-augmented methods, and parameter-efficient fine-tuning into a common evaluation scaffold, motivated by the empirical observation that large pretrained models with simple adaptation can match or surpass specialized meta-learners on many classical FSL distributions. While such unification is necessary for modern practice, it also sharpens the methodological gap we address: if methods differ fundamentally in their adaptation-time resource usage (prompt tokens versus gradient steps versus retrieval), then a unified benchmark that reports only utility necessarily conflates algorithmic progress with increased per-episode compute.

The literature on "compute-aware" or "cost-aware" evaluation in ML provides conceptual tools but is not yet integrated into episodic FSL in a statistically disciplined way. In supervised learning and systems benchmarking, it is increasingly common to report throughput, latency, memory, and energy alongside accuracy, with standardized harnesses and audited rules (e.g., MLPerf-style closed divisions) **?**. In "Green AI" **??**, the community has advocated reporting energy and carbon proxies, and a variety of tools (e.g., experiment trackers and power loggers) implement protocols that translate hardware power samples into Joules and then into CO_2 via carbon intensity and power usage effectiveness (PUE) assumptions **?**. These reporting norms have clarified that FLOPs alone are an incomplete surrogate for environmental and deployment cost, since memory traffic, utilization, batching, and clock/power management can dominate energy at fixed FLOPs. However, most Green AI guidance is phrased at the level of *training runs* or large-scale experiments, not at the level of *episodic adaptation+inference* where the relevant unit is an episode $e = (S, Q, \text{meta})$ and costs can vary substantially with S (e.g., prompt length, retrieval hits, number of adaptation steps). Our contribution is not a new energy model, but a benchmark formulation that treats these costs as episode-level random variables to be estimated and compared with the same care as accuracy.

Statistical methodology for benchmark leaderboards is another area where existing practice falls short of what the modern FSL regime requires. In classical episodic evaluation, it is standard to report a mean accuracy over a finite set of tasks with a standard error, but it is less common to frame ranking claims with explicit (ε, δ) guarantees or to use paired designs as a first-class principle. Yet paired evaluation—running all methods on the identical episode sequence—is precisely the design that reduces ranking variance when episode difficulty is a dominant source of noise. This observation is well-understood in experimental design and in the literature on paired tests and variance reduction, but it is rarely elevated to a benchmark *invariant*. When one adds cost coordinates with measurement noise (e.g., noisy power sampling), unpaired evaluation becomes even harder to interpret, because differences in episode composition can induce apparent cost differences that are unrelated to the method. Accordingly, our protocol insists that both utility and costs be logged per episode under a fixed harness, enabling confidence intervals not only for mean utility but also for frontier functionals and budgeted metrics.

In-context learning introduces robustness and calibration issues that further motivate a frontier-based view. A substantial body of work documents that ICL performance depends sensitively on prompt formatting, demonstration order, label token choices, and the marginal label distribution in the prompt **??**. Calibration methods (e.g., contextual calibration, verbalizer selection, logit bias correction) can improve accuracy and reduce variance across prompts, but they often introduce additional token overhead (extra prompts, multiple forward passes, or auxiliary queries). Similarly, self-consistency, ensembling over permutations, and retrieval-based demonstration selection can be interpreted as mechanisms that trade additional computation and tokens for higher or more stable utility. These phenomena are typically studied by holding one cost axis implicit (e.g., reporting the best accuracy over many prompt variants), which is informative for algorithm design but insufficient for deployment and, in our view, insufficient for benchmark comparison: the act of searching over prompts is itself a cost, and the resulting method should be understood as a set of achievable $(U, \mathbf{C})$ points rather than a single scalar.

We also note connections to multi-objective evaluation in adjacent areas. In reinforcement learning and neural architecture search, Pareto frontiers over reward versus latency/compute are standard, and scalarizations such as hypervolume are used to compare families of tradeoffs. Similarly, in approximate inference and anytime prediction, one reports accuracy as a function of computation budget. These paradigms suggest that the right object to report is not a single operating point but a curve or set indexed by declared knobs. Episodic FSL is particularly well-suited to this view because the benchmark already samples tasks; adding budgets (tokens, FLOPs, energy) yields a natural accuracy-at-budget functional. What has been missing is a benchmark specification that (i) defines the relevant cost coordinates at episode granularity, (ii) enforces a measurement protocol to make those coordinates comparable across methods, and (iii) couples the reporting to finite-sample statistical guarantees.

In summary, existing FSL benchmarks contribute high-quality task distributions and standardized episode generation, and the Green AI literature contributes measurement norms and cost terminology. The ICL robustness literature contributes evidence that adaptation-time knobs materially affect utility. Yet, to our knowledge, there is no end-to-end benchmark formulation that treats episodic few-shot adaptation as inducing a joint distribution over utility and costs, estimates this joint object with controlled uncertainty, and structures leaderboards around Pareto frontiers and budgeted metrics rather than accuracy alone. CarbonBench-FSL is designed to occupy this gap: we preserve the episodic protocol and broadened task distributions of prior work, while formalizing cost measurement and statistical reporting so that comparisons are reproducible, multi-objective, and interpretable under explicit budgets.

3 Formal Setup and Notation

We fix an *episodic* evaluation regime in which performance and resource usage are random variables induced by a task distribution. An *episode* (or task instance) is a tuple

$$e = (S, Q, \mathrm{meta}),$$

where S is a *support* set used for adaptation, Q is a *query* set used for evaluation, and meta denotes auxiliary episode metadata (e.g., domain identifier, modality, class taxonomy, shot/way, or a shift label). We write $\mathcal{Q}$ for a distribution over episodes; our evaluation samples episodes $e_1, \dots, e_n \overset{\text{i.i.d.}}{\sim} \mathcal{Q}$ unless stated otherwise by a benchmark track. The i.i.d. assumption is not a modeling convenience but a benchmark contract: it is the hypothesis under which we will attach (ε, δ)-type guarantees to reported utilities, costs, and derived leaderboard functionals.

The support set S may contain labeled examples, unlabeled examples, or both, depending on the intended adaptation setting. When S is labeled we write $S = \{(x_i, y_i)\}_{i=1}^{|S|}$; when unlabeled components are present we may decompose $S = (S^{\ell}, S^{u})$. Likewise the query set is $Q = \{(x_j, y_j)\}_{j=1}^{|Q|}$ for supervised evaluation (with the y_j held out from the method and used only for scoring). We stress that the episode structure is allowed to vary with meta; for instance, the number of classes, degree of imbalance, or the size of S may be heterogeneous across draws from $\mathcal{Q}$.

We evaluate a finite collection of adaptation methods $\mathcal{M}$. Each $m \in \mathcal{M}$ is required to implement a standardized interface consisting of an *adaptation* phase and an *evaluation* phase. Abstractly, on episode $e = (S, Q, \mathrm{meta})$, the method produces predictions on Q after using S (and optionally meta):

$$\widehat{y}_Q \leftarrow m.\mathrm{Eval}\big(m.\mathrm{Adapt}(S, \mathrm{meta}),\ Q, \mathrm{meta}\big).$$

This abstraction covers in-context learning (where "Adapt" constructs a prompt and "Eval" performs forward passes), gradient-based finetuning or PEFT (where "Adapt" runs updates and "Eval" runs inference), meta-learners (where "Adapt" performs a small inner loop), and hybrids such as retrieval-augmented prompting (where "Adapt" may query an index). Any additional external resources (e.g., retrieval corpora, tools, or extra training data) must be declared as part of the method specification; in the formalism, such resources are viewed as part of the method m rather than part of the episode draw.

For each method m and episode e, we define a *utility* random variable $U_m(e)$ capturing episodic task performance under the benchmark scoring rule. The default utility is query accuracy, but we allow any bounded functional such as macro-F1, reward, or calibration-aware utility (e.g., a proper scoring rule). We assume that the benchmark specifies a measurable map from predictions on Q and ground-truth labels in Q to a scalar in $[0, 1]$:

$$U_m(e) \in [0, 1].$$

The induced randomness in $U_m(e)$ arises from the episode draw $e \sim \mathcal{Q}$ and may also include method-internal randomness (e.g., sampling, dropout at inference, or randomized prompt permutations). When methods are randomized, we interpret $U_m(e)$ as the utility of the method with its prescribed randomness; evaluation fixes the random seed schedule to ensure reproducibility and to support paired comparisons across methods.

Cost-aware evaluation requires that we treat adaptation-time and inference-time resource usage as first-class objects. We therefore associate to each (m, e) a vector of costs

$$\mathbf{C}_m(e) = \big(C_m^{\text{tok}}(e),\, C_m^{\text{flop}}(e),\, C_m^{\text{param}}(e),\, C_m^{\text{comm}}(e),\, C_m^{\text{J}}(e),\, C_m^{\text{CO}_2}(e)\big),$$

where C_m^{tok} is the number of prompt/context tokens consumed at adaptation+inference, C_m^{flop} is an estimated FLOP count for the same computation, C_m^{param} is the number of parameters (or effective degrees of freedom) updated during adaptation, C_m^{comm} is an optional communication cost for federated or client–server variants, C_m^{J} is energy in Joules, and $C_m^{\text{CO}_2}$ is a CO_2 proxy derived from energy and carbon-intensity/PUE assumptions. The benchmark may choose a subset of these coordinates as mandatory, but the evaluation harness must define each selected coordinate precisely and compute it consistently across all methods.

We distinguish between *intrinsic* costs (e.g., token counts and FLOPs estimates) and *measured* costs (e.g., time and energy). Intrinsic costs are computed deterministically from method traces under a declared tokenizer and FLOP accounting rules. Measured costs depend on a hardware and systems profile HW, which fixes device type (GPU/CPU), driver/runtime versions, power sampling method, batching rules, warmup protocol, and any normalization choices such as power usage effectiveness (PUE) and carbon intensity. Formally, we treat HW as part of the experimental condition and write the measured coordinates as conditional random variables given HW. In particular, $C_m^{\text{J}}(e)$ is defined as the energy attributable to the adaptation+inference workload for episode e executed under HW and the benchmark protocol.

Because energy and related costs are obtained from instrumentation, we explicitly model measurement noise. Let $\widetilde{\mathbf{C}}_m(e)$ denote the observable cost vector returned by the harness for method m on episode e. We posit an additive noise model

$$\widetilde{\mathbf{C}}_m(e) = \mathbf{C}_m(e) + \boldsymbol{\eta}_{m,e},$$

where $\boldsymbol{\eta}_{m,e}$ is a zero-mean noise vector induced by sampling resolution, clock/power management, background processes allowed by the protocol, and estimator error in converting power samples to energy. The protocol is designed so that $\mathbb{E}[\boldsymbol{\eta}_{m,e} \mid e] = \mathbf{0}$ and each coordinate is either bounded or sub-Gaussian with a known (or conservatively bounded) variance proxy. This assumption is again a benchmark contract: it encodes what it means for the measurement harness to be *standardized.* When repeated runs are permitted, we may reduce noise by averaging over repeats; otherwise, the statistical analysis treats $\widetilde{\mathbf{C}}_m(e)$ as a noisy but unbiased observation of $\mathbf{C}_m(e)$.

Finally, we emphasize that *paired* evaluation is built into our notation. Since all methods are run on the identical episode sequence $e_1, \ldots, e_n$, we can meaningfully consider episode-wise differences (in utility or cost) and exploit correlation across methods induced by shared episode difficulty. This pairing will be central when we later define ranking stability and confidence intervals for derived functionals. The present section merely fixes the objects: a task distribution $\mathcal{Q}$ over episodes e, a finite method set $\mathcal{M}$ with a standardized adapt–evaluate interface, a bounded utility $U_m(e)$, a cost vector $\mathbf{C}_m(e)$ measured under a declared HW, and a noise model for the observed costs returned by the harness.

4 Problem Definition: Cost-Aware Few-Shot Evaluation (CAFSE)

We now formalize the benchmark problem induced by the objects fixed in §3. The central design choice is that we do not regard a method as producing a single number, but rather a *utility–cost tradeoff* that can be queried at different adaptation "knobs" (prompt length, update steps, adapter rank, communication rounds, *etc.*). CAFSE is thus a multi-objective evaluation problem with statistical guarantees, where the primary output is a Pareto frontier and any scalar leaderboard is a derived functional of that frontier.

Methods with declared knob sets. For each $m \in \mathcal{M}$ we assume a declared (finite or discretized) configuration set Θ_m of admissible knob settings. A configuration $\theta \in \Theta_m$ fixes all budget-relevant choices of the method (e.g., number of gradient steps, batch size, context length, retrieval depth, precision mode), as well as any stopping rule parameters that affect cost. Executing m on an episode e under configuration θ yields random utility and cost observations

$$U_{m,\theta}(e) \in [0,1], \qquad \mathbf{C}_{m,\theta}(e) \in \mathbb{R}_+^{d_C},$$

where $\mathbf{C}$ may include intrinsic coordinates and measured coordinates under HW. In all tracks, the benchmark contract requires that the mapping $(m, \theta, e) \mapsto (U_{m,\theta}(e), \widetilde{\mathbf{C}}_{m,\theta}(e))$ is executed by a standardized harness with fixed warmup, batching, and instrumentation rules, so that comparisons across methods are meaningful.

Input and observable data. CAFSE takes as input (i) an episode sampler for $\mathcal{Q}$ producing i.i.d. episodes $e_1, \ldots, e_n$, (ii) a finite set of methods $\mathcal{M}$ with configuration sets Θ_m and a standardized adapt–evaluate interface, (iii) a metrics specification that fixes the utility functional U and the coordinates of $\mathbf{C}$ to be reported, and (iv) a hardware and systems profile HW determining the measurement protocol for energy/CO$_2$ proxies. The observable log for a fixed (m, θ) consists of paired samples

$$\mathcal{L}_{m,\theta} = \big\{(u_{m,\theta,t}, \, \widetilde{\mathbf{c}}_{m,\theta,t})\big\}_{t=1}^{n}, \quad u_{m,\theta,t} = U_{m,\theta}(e_t), \quad \widetilde{\mathbf{c}}_{m,\theta,t} = \widetilde{\mathbf{C}}_{m,\theta}(e_t),$$

where $\widetilde{\mathbf{C}}$ denotes the harness-observed costs (potentially noisy but unbiased under the protocol assumptions). The paired design is enforced by construction: the *same* episode sequence $\{e_t\}_{t=1}^{n}$ is used for all methods and all reported configurations.

Primary statistical targets. For each fixed (m, θ), CAFSE targets the episode-marginal expectations

$$\mu_{m,\theta} := \mathbb{E}_{e\sim\mathcal{Q}}\big[U_{m,\theta}(e)\big], \qquad \boldsymbol{\nu}_{m,\theta} := \mathbb{E}_{e\sim\mathcal{Q}}\big[\mathbf{C}_{m,\theta}(e)\big],$$

together with uncertainty quantification (confidence intervals or, more generally, (ε, δ)-accurate bounds). The benchmark may require reporting either marginal means per coordinate or a small number of scalarizations (e.g., Joules and tokens). When costs are measured with additive noise, $\boldsymbol{\nu}_{m,\theta}$ is interpreted as the latent mean cost and the empirical estimator is formed from $\widetilde{\mathbf{C}}$.

Frontier as the main output. The CAFSE output for method m is an estimate of the achievable utility–cost tradeoff over Θ_m. We define the *(expected) Pareto frontier* as the set of configurations not dominated in expectation:

$$\mathcal{F}_m := \Big\{(\boldsymbol{\nu}_{m,\theta}, \mu_{m,\theta}) : \theta \in \Theta_m, \ \nexists\, \theta' \in \Theta_m \text{ s.t. } \boldsymbol{\nu}_{m,\theta'} \le \boldsymbol{\nu}_{m,\theta} \text{ and } \mu_{m,\theta'} \ge \mu_{m,\theta}, \text{ with at least one strict}$$

where inequalities over cost vectors are coordinate-wise. The empirical analogue $\widehat{\mathcal{F}}_m$ is computed from plug-in estimators

$$\widehat{\mu}_{m,\theta} = \frac{1}{n}\sum_{t=1}^{n} u_{m,\theta,t}, \qquad \widehat{\boldsymbol{\nu}}_{m,\theta} = \frac{1}{n}\sum_{t=1}^{n} \widetilde{\mathbf{c}}_{m,\theta,t},$$

optionally augmented with confidence sets per point. We treat the frontier, not a scalar score, as the benchmark's principal scientific artifact: it exposes whether gains in U require disproportionate increases in tokens, FLOPs, energy, or communication.

Budgeted performance functionals. Leaderboards typically demand scalars. In CAFSE, any scalar score must be a declared functional of the frontier. We emphasize three canonical classes.

Accuracy-at-budget. For a scalar budget B and a selected cost coordinate (or scalarized cost) C, define

$$\mathrm{Acc@}B(m) := \sup_{\theta\in\Theta_m:\ \nu^{(C)}_{m,\theta}\le B} \mu_{m,\theta}, \qquad \widehat{\mathrm{Acc@}B}(m) := \max_{\theta\in\Theta_m:\ \widehat{\nu}^{(C)}_{m,\theta}\le B} \widehat{\mu}_{m,\theta},$$

with the convention that infeasible methods return $-\infty$ (or are excluded) if no configuration meets the budget. Vector budgets are handled analogously with coordinate-wise constraints.

Area under the frontier (AUF). Over a budget interval $[B_{\min}, B_{\max}]$, we may define the utility envelope

$$f_m(B) := \sup_{\theta:\ \nu^{(C)}_{m,\theta}\le B} \mu_{m,\theta}, \qquad \mathrm{AUF}(m) := \frac{1}{B_{\max}-B_{\min}}\int_{B_{\min}}^{B_{\max}} f_m(B)\,dB,$$

with $\widehat{f}_m$ and $\widehat{\mathrm{AUF}}(m)$ computed from $\widehat{\mathcal{F}}_m$ by piecewise-constant interpolation over the discretized configurations.

Hypervolume. In two dimensions (one cost coordinate and utility), after fixing a reference point $(B_{\mathrm{ref}}, U_{\mathrm{ref}})$, we may score the dominated region induced by the frontier; this is a standard choice when one wishes to reward uniformly good tradeoffs rather than peak performance at a single budget.

Rankings with error control. CAFSE returns (i) frontier estimates with uncertainty, (ii) chosen scalar scores with uncertainty, and (iii) a ranking induced by these scores. Since rankings are unstable under noise, the benchmark must specify tie-handling. Concretely, when confidence intervals overlap, we treat methods as statistically indistinguishable at the stated confidence level; when multiple comparisons are performed, the harness applies a declared correction (e.g., Holm–Bonferroni) or reports families of non-dominated methods under the uncertainty sets.

Desiderata and benchmark contracts. We require the following properties as part of the CAFSE problem definition.

- *Fairness (paired, budget-consistent evaluation).* All methods are evaluated on identical episode draws and identical splits (S, Q), with identical budget definitions and stopping criteria. Any external data, retrieval corpus, tool access, or extra training is declared and attributed to the appropriate track.
- *Reproducibility (deterministic harness semantics).* The evaluation harness fixes tokenizer versions, FLOP accounting rules, and HW (including sampling resolution and warmup). Randomness is controlled by a logged seed schedule; repeated-run variance estimation is permitted only when uniformly available.
- *Statistical validity (finite-sample uncertainty).* Reported means and derived frontier functionals are accompanied by confidence intervals under stated assumptions (boundedness or sub-Gaussian noise, i.i.d. episodes). The episode budget n is treated as a first-class resource that determines attainable (ε, δ) guarantees.
- *Cost awareness (multi-objective outputs).* Methods are not compared solely by utility; Pareto frontiers (and, when desired, budgeted scalarizations) are mandatory outputs, ensuring that improvements in U are contextualized by their resource implications.

Under these contracts, CAFSE provides a precise target for the evaluation algorithm: from paired episode logs, estimate per-method utility–cost frontiers and any declared scalar leaderboard functionals with quantified uncertainty, enabling meaningful comparison under explicit deployment-relevant budgets.

5 CarbonBench-FSL Benchmark Suite Specification

We next specify the benchmark suite underlying CarbonBench-FSL. The suite is not a single dataset, but a family of episodic task distributions obtained by combining (i) *task families* (classification, structured prediction, generation, decision-making) across (ii) *modalities* (text, vision, audio, multimodal), and (iii) *shift regimes* (in-domain, cross-domain, and related forms of distribution shift). The outcome of this specification is a collection of episode samplers, each defining a distribution $\mathcal{Q}$ over episodes $e = (S, Q, \mathrm{meta})$ with mandatory, method-independent splits.

Task families and episode templates. We fix a finite index set of families $\mathcal{D}$, where each $d \in \mathcal{D}$ provides a base population of instances $\mathcal{X}_d$ together with a labeling or reward mechanism. An *episode template* specifies how to construct S and Q from $\mathcal{X}_d$ and determines the semantics of utility U for that family. Concretely, each family d defines a distribution $\mathcal{Q}_d$ by the following generic procedure:

1. Sample episode metadata meta, which may include a domain identifier, label set, language, prompt template identifier, and difficulty parameters.
2. Sample (or induce) a task instance τ (e.g., a class subset, a relation type, a program, a goal) from a family-specific task prior.
3. Draw a support set S and query set Q conditionally on τ and meta, with the constraint $S \cap Q = \emptyset$ at the instance level.

For few-shot classification families, we use an N-way, K-shot template with fixed (N, K) per benchmark track, and a query size $|Q|$ chosen to yield stable per-episode utility estimates. For structured prediction (e.g., tagging) and generation (e.g., short-form QA), S contains a fixed number of supervised exemplars and Q contains held-out instances scored by exact match, token-level F1, or a declared task reward. For decision-making families, S may contain demonstrations or offline trajectories and Q corresponds to evaluation rollouts or held-out contexts, with U defined as episodic return under an evaluation environment specified in meta. The suite is agnostic to the presence of unlabeled data: if provided, it is included explicitly as an episode component and used uniformly across methods.

Shift regimes as first-class benchmark axes. To evaluate adaptation methods beyond i.i.d. in-domain generalization, we define regimes as separate distributions (or sub-distributions) over episodes. For each family d, we specify a set of regimes $\mathcal{R}_d$ and associated samplers $\{\mathcal{Q}_{d,r}\}_{r \in \mathcal{R}_d}$. At minimum, we include:

- *In-domain (ID).* Episodes are drawn from a single domain and a fixed preprocessing pipeline; meta-train and meta-test differ only through episode resampling, not through domain shift.
- *Cross-domain (CD).* Episodes are drawn from multiple domains with a designated split into *source* domains (available for method development within the benchmark) and *target* domains (held out for evaluation). Formally, if domains are indexed by $z \in \mathcal{Z}_d$, then $\mathcal{Z}_d = \mathcal{Z}_d^{\text{src}} \cup \mathcal{Z}_d^{\text{tgt}}$ with $\mathcal{Z}_d^{\text{src}} \cap \mathcal{Z}_d^{\text{tgt}} = \emptyset$, and the test-time sampler places mass only on $\mathcal{Z}_d^{\text{tgt}}$.
- *Cross-label-set / cross-task (CL).* The label sets or task identifiers sampled in meta are disjoint across development and test partitions. This is the canonical setting for few-shot class generalization in vision and entity/relation generalization in NLP.

Where applicable, we optionally refine the above by language shift (cross-lingual), style shift (register or genre), and context shift (different prompt templates), each implemented as an explicit component of meta and a disjointness constraint between development and test identifiers. The benchmark treats each (d, r) as a distinct evaluation condition; overall reporting may average across conditions or present per-condition frontiers.

Modalities and unified episodic interface. CarbonBench-FSL includes modality-specific families but enforces a single episodic contract. In text families, instances are sequences paired with labels or targets; in vision families, instances are images with labels; in audio families, instances are waveforms or spectrograms; and in multimodal families, instances are aligned tuples (e.g., image–text or audio–text). In all cases, the episode is serialized for methods through the same abstract fields (support/query plus metadata), while permitting modality-specific renderings (e.g., image tensors versus image URLs) as long as all methods in a track receive identical inputs. We regard multimodal prompting and tool use as track-level decisions: if a track allows external tools (OCR, ASR, retrieval), then the access pattern is declared and applied uniformly; otherwise such access is disallowed to preserve comparability.

Mandatory dataset and episode splits. We impose mandatory splits at two levels: *instance-level* partitions and *task-identifier* partitions. Instance-level partitions ensure that, within each (d, r), there is no leakage of individual examples between development and test samplers. Task-identifier partitions ensure that, when the regime calls for it (e.g., CL), the identifiers that determine the task—such as classes, relations, domains, languages, or templates—are disjoint across partitions.

Concretely, for each family d we define three benchmark partitions: *development* (public), *test* (held-out), and an optional *diagnostic* (public) split for ablations. The development split supports method debugging and knob selection; the official test split is used for reported scores. The sampler for the test split is fixed (including its random seed schedule) by the benchmark organizers, so that all submissions are evaluated on an identical episode sequence. Within each episode, we enforce disjointness $S \cap Q = \emptyset$ and fix $|S|, |Q|$ by track; any deviation (e.g., variable-shot episodes) must be declared as a separate subtrack so that budget comparisons remain interpretable.

Closed versus open data usage. Because adaptation methods differ in their reliance on external corpora (retrieval, continued pretraining, instruction data), the suite defines data-usage policies as separate tracks. In a *closed* track, only the benchmark-provided development split may be used for any training or retrieval corpus construction beyond the pretrained model itself. In an *open* track, additional data may be used, but must be declared with provenance and licensing, and is treated as part of the method description rather than the benchmark definition. In either case, the *test* episode sampler remains fixed and unseen during development.

Optional track structure: federated variants. We include an optional federated track in which each episode is augmented with a client partition. Formally, an episode becomes $e = (\{S^{(i)}\}_{i=1}^{M}, Q, \text{meta})$ where $S^{(i)}$ is client i's local support set and Q is a global (or per-client) query set specified by meta. The track specifies whether adaptation is (i) local-only with no communication, (ii) server-mediated with a bounded number of rounds, or (iii) hybrid (e.g., local adapters plus occasional aggregation). The benchmark does not prescribe a particular federated algorithm; it prescribes only the partitioning scheme (including heterogeneity patterns across clients) and the episode interface, leaving communication and coordination to the method.

Optional track structure: continual variants. We also include an optional continual adaptation track in which the evaluation oracle provides a *sequence* of episodes $(e_t)_{t=1}^{n}$ with nonstationary metadata (e.g., a domain index that drifts over time). Methods may maintain state across episodes, subject to declared memory constraints and the same knob semantics as in the stationary setting. The suite specification for this track consists solely of (i) the drift schedule encoded in meta_t, (ii) the rule determining what information from past episodes may be retained (e.g., bounded replay buffer, bounded parameter updates), and (iii) the requirement that evaluation utilities are computed on the provided queries without retroactive access to future episodes. We treat this track as optional because it introduces additional modeling choices, but we include its interface so that cost-aware adaptation can be studied under realistic nonstationarity.

6 Measurement Protocol

We next fix a measurement protocol whose role is to convert a method execution on an episode into a standardized cost vector $\mathbf{C}_m(e) = (C_m^{\text{tok}}(e), C_m^{\text{flop}}(e), C_m^{\text{param}}(e), C_m^{\text{comm}}(e), C_m^{\text{J}}$ together with auxiliary observables (wall-clock time, power traces) that permit calibration of measurement noise and reproducible re-evaluation. The protocol is part of the benchmark definition: submissions are required to expose the hooks needed for instrumentation, and the evaluation harness enforces uniform warmup/batching rules under a fixed hardware profile HW.

Phase separation and cost attribution. For each method m and episode $e = (S, Q, \text{meta})$, we separate execution into *adaptation* (any computation that depends on S and produces an episode-specific state) and *inference* (prediction on Q given the adapted state). Costs are recorded for the union of these phases, and, when supported by the submission, also reported as a decomposition $\mathbf{C}_m(e) = \mathbf{C}_m^{\text{adapt}}(e) + \mathbf{C}_m^{\text{infer}}(e)$. This separation is particularly important for methods with nontrivial adaptation (gradient steps, retrieval indexing, local training in federated variants), while remaining well-defined for in-context learning (ICL), for which the adaptation component may be vacuous.

Token accounting. We define $C_m^{\text{tok}}(e)$ as the total number of model tokens processed *at test time* to produce predictions for the entire query set Q, including both input and generated output tokens. For text-only models, this is measured by the model tokenizer used in the submission. For multimodal models, we require a declared tokenization scheme for non-text inputs (e.g., image patch tokens or audio frame tokens) so that counts are comparable within a track. For ICL, the input tokens include all fixed prompt text (system instructions), all serialized support exemplars, and all query prompts; output tokens include any generated rationales if the method chooses to produce them. For retrieval-augmented methods, tokens consumed by the generator are counted as usual; in addition, any token-level interaction with a learned retriever that uses a language-model forward pass (e.g., bi-encoder scoring via the same backbone) must be included in $C_m^{\text{tok}}(e)$, while purely classical retrieval is accounted for in $C_m^{\text{flop}}(e)$ and wall-clock time. The harness records token counts by intercepting model calls; submissions that batch multiple query items must still report aggregate tokens over the batch.

FLOPs and updated-parameter accounting. We define $C_m^{\text{flop}}(e)$ as an estimate of floating-point operations for the adaptation and inference phases combined. Because exact FLOPs are hardware- and kernel-dependent, we standardize the estimator: the harness computes FLOPs from an agreed-upon per-layer analytical model for the declared architecture, using observed sequence lengths and batch sizes, and multiplies by the number of forward/backward passes actually executed. If a submission provides profiler-based FLOPs (e.g., from a framework tracer), we log both estimates and require agreement within a stated tolerance on a calibration suite; otherwise, the analytical estimator is authoritative for leaderboard costs. We define $C_m^{\text{param}}(e)$ as the number of parameters updated during adaptation (or, more generally, the number of effective degrees of freedom trained). For full finetuning, this is the full parameter count of the adapted model; for PEFT, it is the count of trainable adapter parameters (e.g., LoRA matrices) plus any trainable layer norms or embeddings; for ICL and frozen-feature methods, $C_m^{\text{param}}(e) = 0$. For methods that update only a subset of parameters conditionally on meta, the submission must expose the updated set so that $C_m^{\text{param}}(e)$ is computed episode-wise.

Wall-clock time and batching invariants. We record wall-clock time as an auxiliary scalar $T_m(e)$ measured from the start of adaptation to the end of query prediction, excluding dataset loading that is common across methods but including any method-specific preprocessing (serialization, retrieval index access, client–server synchronization). To reduce variance from kernel compilation and caching, the harness enforces: (i) a fixed number of warmup executions per method per process; (ii) a fixed batching policy (maximum batch size or maximum tokens per batch) declared by the track; and (iii) a fixed precision regime (e.g., FP16/BF16) unless the track explicitly permits alternatives. These invariants ensure that $T_m(e)$ and energy measurements are comparable and that token/FLOP estimates correspond to realized execution.

Communication cost for federated variants. For federated tracks, we define $C_m^{\text{comm}}(e)$ as a vector (logged but scalarized by the leaderboard when needed) consisting of total bytes transmitted and the number of communication rounds. Concretely, if the episode induces messages $\{\mathsf{msg}_{r,i\to j}\}$ over rounds r, then $C^{\text{comm}}_{m,\text{bytes}}(e) = \sum_r \sum_{i\to j} \text{size}(\mathsf{msg}_{r,i\to j})$, and $C^{\text{comm}}_{m,\text{rounds}}(e)$ is the number of synchronized round barriers. The submission must mark message boundaries so that compression, quantization, and sparsification are reflected in the byte count rather than obscured by implementation artifacts.

Energy measurement in Joules. We define $C_m^{\text{J}}(e)$ as measured energy (Joules) attributable to the method execution for episode e under HW. The harness collects a time series of instantaneous power $P(t)$ from standardized sensors (e.g., GPU power via NVML, CPU package power via RAPL) at a fixed sampling cadence, and integrates over the measured interval: $C_m^{\text{J}}(e) = \int_{t_{\text{start}}}^{t_{\text{end}}} \big(P(t) - P_{\text{idle}}\big)\, dt$, where P_{idle} is an empirically estimated idle baseline for the same process placement and temperature regime. We require (a) pinning to a fixed device set, (b) disabling frequency scaling when feasible or logging the frequency governors, and (c) logging sensor resolution and sampling rate. When only energy counters are available (rather than power samples), we use counter differences over the same interval and still subtract an idle estimate. In all cases, we log raw traces so that post hoc audits can recompute $C_m^{\text{J}}(e)$ under alternative baseline assumptions.

CO_2 proxy. We define $C_m^{\text{CO}_2}(e)$ as an operational emissions proxy derived from measured energy and declared facility assumptions:

$$C_m^{\text{CO}_2}(e) = \frac{C_m^{\text{J}}(e)}{3.6 \times 10^6} \cdot \text{CI} \cdot \text{PUE},$$

where CI is the carbon intensity in $\text{gCO}_2\text{e/kWh}$ and $\text{PUE} \geq 1$ is a power-usage effectiveness factor. The benchmark fixes default values of CI and PUE per HW profile (or requires the evaluator to declare them), and we report sensitivity by optionally providing $C_m^{\mathrm{CO}_2}(e)$ under a small set of canonical $(\mathrm{CI}, \mathrm{PUE})$ pairs.

Noise calibration and run-to-run variance. Energy and time measurements exhibit nontrivial noise (sensor quantization, background system activity, thermal throttling). We therefore incorporate a calibration phase in which the harness (i) measures idle power and its variance, (ii) runs a fixed microbenchmark at several durations to estimate integration error, and (iii) optionally repeats a subset of episodes to estimate run-to-run variance of $(U_m(e), \mathbf{C}_m(e))$. For repeated runs, we treat the observed costs as noisy but (approximately) unbiased and record the empirical variance; these quantities feed directly into the confidence intervals used later. Submissions that are intrinsically stochastic (sampling-based decoding, randomized adaptation) must expose random seeds so that the harness can distinguish algorithmic variance from measurement variance.

Reproducibility invariants and logging. Finally, we fix a minimal set of invariants required for reproducibility: versioned code and model identifiers; exact tokenizer and prompt serialization; declared knob settings (prompt length, adapter rank, steps, batch size, rounds); deterministic episode ordering; and a complete record of HW (device model, driver/runtime versions, precision mode, sensor type, and CI/PUE assumptions). The harness emits a per-episode log entry containing $(u_{m,t}, \mathbf{c}_{m,t}, T_{m,t})$ together with hashes of inputs/outputs sufficient to verify that two evaluations are semantically identical. Under these invariants, the remaining discrepancies in $\mathbf{C}$ are attributable to bounded measurement noise, which is explicitly quantified and propagated to uncertainty estimates.

7 Leaderboard Metrics

The benchmark output is multi-objective: for each method $m \in \mathcal{M}$ and episode $e \sim \mathcal{Q}$ we observe a utility $U_m(e)$ together with a cost vector $\mathbf{C}_m(e)$. A leaderboard must therefore specify (i) which tradeoffs are considered admissible (Pareto efficiency), (ii) which scalar summaries are reported for ease of ranking, and (iii) how ties and stochasticity are handled so that the reported ordering is stable under re-evaluation.

Knobs and the induced set of achievable points. Each method m declares a (finite or discretized) set of knob settings Θ_m controlling its resource–accuracy tradeoff (e.g., prompt length, number of gradient steps, adapter rank, number of communication rounds). Running m with knob $\theta \in \Theta_m$ on episode e yields random variables $U_{m,\theta}(e)$ and $\mathbf{C}_{m,\theta}(e)$. We emphasize that

θ is part of the method definition for evaluation purposes: if a submission adaptively chooses θ episode-wise, then θ is determined by a declared policy π_m and the resulting method is evaluated as $m \circ \pi_m$, with costs attributed to the realized choice.

Pareto dominance and frontier definition. We adopt the convention that higher utility is better and lower costs are better (coordinate-wise). For two knob settings $\theta, \theta' \in \Theta_m$, we say that θ *dominates* θ' (write $\theta \succeq \theta'$) if

$$\mathbb{E}[U_{m,\theta}] \geq \mathbb{E}[U_{m,\theta'}] \qquad \text{and} \qquad \mathbb{E}[\mathbf{C}_{m,\theta}] \leq \mathbb{E}[\mathbf{C}_{m,\theta'}]$$

coordinate-wise, with at least one inequality strict. The (population) Pareto set of knob settings is

$$\mathcal{P}_m \;=\; \big\{\theta \in \Theta_m : \nexists\, \theta' \in \Theta_m \text{ with } \theta' \succeq \theta\big\},$$

and the corresponding utility–cost *frontier* is the image

$$\mathcal{F}_m \;=\; \big\{\big(\mathbb{E}[U_{m,\theta}], \mathbb{E}[\mathbf{C}_{m,\theta}]\big) : \theta \in \mathcal{P}_m\big\}.$$

Because $\mathbf{C}$ is multi-dimensional, $\mathcal{F}_m$ is generally a set in $[0,1] \times \mathbb{R}_+^{d_C}$. The benchmark reports $\widehat{\mathcal{F}}_m$ estimated from finite episodes and, for visualization, also reports two-dimensional projections such as $(\mathbb{E}[U], \mathbb{E}[C^{\mathrm{J}}])$ and $(\mathbb{E}[U], \mathbb{E}[C^{\mathrm{tok}}])$, which are interpretable deployment axes.

Budgeted metrics: accuracy at budget and cost at accuracy. While the frontier is the primary object, many users require a scalar summary at a fixed budget. For a scalar cost coordinate C (typically C^{J}, C^{tok}, or a scalarization of $\mathbf{C}$), and for a budget $B > 0$, we define the *accuracy-at-budget* functional

$$\mathrm{Acc@}B(m; C) \;=\; \sup_{\theta \in \Theta_m :\, \mathbb{E}[C_{m,\theta}] \leq B} \mathbb{E}[U_{m,\theta}],$$

with the convention $\mathrm{Acc@}B = 0$ if the feasible set is empty. Dually, for a target utility level $u \in [0,1]$, we define the *cost-at-accuracy* functional

$$\mathrm{Cost@}u(m; C) \;=\; \inf_{\theta \in \Theta_m :\, \mathbb{E}[U_{m,\theta}] \geq u} \mathbb{E}[C_{m,\theta}],$$

with $\mathrm{Cost@}u = +\infty$ if no knob setting attains u. These are evaluated on a fixed, published set of budgets $\mathcal{B}$ and target utilities $\mathcal{U}$ so that methods with different knob granularities remain comparable. When $\mathbf{C}$ is vector-valued, the track must specify a feasible region (e.g., $C^{\mathrm{tok}} \leq B_{\mathrm{tok}}$ and $C^{\mathrm{J}} \leq B_{\mathrm{J}}$); the corresponding $\mathrm{Acc@}B$ is then defined with $\mathbb{E}[\mathbf{C}_{m,\theta}] \leq \mathbf{B}$ coordinate-wise.

Scalar frontier summaries: hypervolume and AUF. To rank methods without selecting a single operating point, we report functionals of the frontier. In the common two-dimensional case $(\mathbb{E}[U], \mathbb{E}[C])$ where C is a scalar cost, we define an *area under the frontier* (AUF) over a bounded budget interval $[B_{\min}, B_{\max}]$ by

$$\mathrm{AUF}(m; C) \;=\; \frac{1}{B_{\max} - B_{\min}} \int_{B_{\min}}^{B_{\max}} \mathrm{Acc@}B(m; C)\, dB,$$

where Acc@B is interpreted as the upper envelope of achievable utility as a function of budget. We also report a (normalized) *hypervolume* score in (C, U)-space relative to a reference point $(C_{\mathrm{ref}}, U_{\mathrm{ref}})$ chosen by the benchmark:

$$\mathrm{HV}(m; C) \;=\; \lambda\Big(\big\{(c, u) : c \le C_{\mathrm{ref}},\; u \ge U_{\mathrm{ref}}\big\} \cap \mathrm{Dom}(\mathcal{F}_m)\Big),$$

where λ denotes Lebesgue measure and $\mathrm{Dom}(\mathcal{F}_m)$ is the region dominated by the frontier (low cost, high utility). For multi-cost settings, we either (i) compute hypervolume in a low-dimensional projection mandated by the track, or (ii) scalarize costs via a published monotone map $s(\mathbf{C})$ (e.g., $s(\mathbf{C}) = \alpha C^{\mathrm{J}} + (1 - \alpha) C^{\mathrm{tok}}$), and apply the two-dimensional definitions to $(s(\mathbf{C}), U)$. In all cases, the benchmark reports the reference point and normalization constants so that scores are interpretable and reproducible.

Dominance rules and tie-breaking under uncertainty. Because $\mathcal{F}_m$ and scalar metrics are estimated from finite episodes, we avoid deterministic tie-breaking rules that are unstable to sampling noise. The harness therefore reports confidence intervals for each scalar leaderboard metric and uses the following principle: method m_1 is ranked above m_2 only when the lower confidence bound for the chosen score of m_1 exceeds the upper confidence bound for m_2 (or, for cost-minimization metrics, when the upper bound of m_1 is below the lower bound of m_2). Otherwise, the methods are reported as statistically indistinguishable at the declared confidence level and may share a rank interval. For Pareto dominance, we similarly declare that an estimated point $(\widehat{u}_1, \widehat{\mathbf{c}}_1)$ dominates $(\widehat{u}_2, \widehat{\mathbf{c}}_2)$ only if the corresponding one-sided confidence bounds certify domination simultaneously across utility and all reported cost coordinates. This rule prevents spurious dominance claims driven by measurement noise in C^{J} or stochastic decoding variance.

Stochastic methods and prompt sensitivity (ICL). Many methods are intrinsically stochastic: decoding may sample, adaptation may use random initialization, and ICL may depend on the order and selection of demonstrations. We treat such randomness as part of the method and define leaderboard quantities in terms of expectations over both episodes and method randomness. Concretely, a submission must expose a seedable randomness interface; the harness fixes a seed schedule and, when requested by the track, repeats a subset of episodes to estimate the additional variance contribution. For ICL specifically, we distinguish (i) *prompt construction policy* (how demonstrations are chosen and ordered) and (ii) *prompt budget* (how many tokens are allocated). Prompt construction is treated as a declared algorithmic component; if it uses a retrieval corpus or heuristic, this must be declared and evaluated under the appropriate track rules. Prompt sensitivity is then reflected as variance in $U_{m,\theta}(e)$ and $C^{\text{tok}}_{m,\theta}(e)$, and the reported frontier either averages over prompt randomness (default) or, in robustness-focused tracks, additionally reports quantiles (e.g., 10%-worst utility at budget). In all cases, the same prompt policy and knob definitions are used across episodes so that differences in frontier shape correspond to method behavior rather than undisclosed prompt engineering.

8 8. Evaluation Algorithm: pseudocode for sampling episodes, running methods, recording costs, computing estimators, confidence intervals, and frontier metrics; best practices for caching, warmup, and batching without bias.

We now specify the evaluation algorithm executed by the harness. Our goal is to obtain, for each submitted method $m \in \mathcal{M}$ (possibly equipped with a knob $\theta \in \Theta_m$), a paired collection of utility–cost observations over i.i.d. episodes $e_t \sim \mathcal{Q}$, from which we compute (i) mean utility and mean costs with confidence intervals, and (ii) frontier-based leaderboard functionals (e.g., Acc@B, AUF, hypervolume) with uncertainty quantification.

Episode sampling and paired design. We fix a public episode sampler for $\mathcal{Q}$ and draw an episode sequence $e_1, \dots, e_n$ once per evaluation run. Each episode $e_t = (S_t, Q_t, \mathrm{meta}_t)$ contains all information required by the track (e.g., class identities, domain tags, optional unlabeled pool). The harness evaluates every method on the *same* episode sequence (paired evaluation), and, when a knob sweep is required, on the same episode sequence for each $\theta \in \Theta_m$. This pairing is not merely a convenience: it is the design choice that enables low-variance comparisons of methods through episode-wise differences, and it eliminates a major source of irreproducibility stemming from different episode draws.

Standardized method interface and randomness control. Each method m implements two calls: $\mathrm{adapt}_m(S; \theta, \mathrm{seed})$ and $\mathrm{evaluate}_m(Q; \theta, \mathrm{seed})$, with the convention that adapt may be a no-op (e.g., for pure in-context learning) and that evaluate returns both predictions and any method-specific artifacts required for auditing (e.g., selected demonstrations, retrieved documents, or adapter checkpoints). We treat all internal randomness (prompt sampling, parameter initialization, dropout, decoding stochasticity) as part of the method distribution; accordingly, the harness provides a declared seed schedule seed(m, t, θ, r) where r indexes optional repeated runs. If the track requests repeated runs, we repeat only a prespecified subset of episodes to estimate variance contributions without multiplying total cost prohibitively; the subset and repetition count are fixed ex ante and are identical across methods.

Cost measurement and instrumentation hooks. For each execution of adapt + evaluate we record a utility $u_{m,\theta,t}$ and a vector of costs $\mathbf{c}_{m,\theta,t}$. Token costs C^{tok} are computed by a canonical tokenizer and include (i) all prompt/context tokens provided to the base model, (ii) generated output tokens, and (iii) any additional model calls induced by retrieval, self-consistency, verification, or multi-pass decoding. FLOPs costs C^{flop} are reported either from vendor counters when available or from a published estimator calibrated to HW. Parameter-update costs C^{param} count the effective number of degrees of freedom updated during adaptation (e.g., all weights for full finetuning, LoRA parameters for PEFT). For federated or client–server variants we additionally log communication C^{comm} as bytes and/or rounds, with the protocol defining what metadata is included. Energy C^{J} is measured via standardized power sampling integrated over the adapt+eval window, with the harness controlling sampling rate, synchronization points, and inclusion/exclusion boundaries; CO_2 proxies C^{CO_2} are derived from energy using declared carbon intensity and PUE assumptions attached to HW. Because these measurements can be noisy, we log raw samples (or sufficient statistics) so that auditors can recompute integrals and verify segmentation.

Pseudocode and logging discipline. The harness maintains per-method logs $\mathcal{L}_{m,\theta} = \{(u_{m,\theta,t}, \mathbf{c}_{m,\theta,t})\}_{t=1}^{n}$. Conceptually, the evaluation proceeds as follows: for each episode t, we materialize S_t, Q_t, then for each method m (and each θ when sweeping) we execute adapt+eval under the fixed seed schedule, record $u_{m,\theta,t}$, and record $\mathbf{c}_{m,\theta,t}$ under the measurement protocol. We require that all logged values be attached to the episode identifier and the knob setting, and that failures (timeouts, OOM, invalid outputs) be logged explicitly and handled by a track-defined rule (e.g., utility 0 with costs measured up to failure, or exclusion with a recorded failure count). This explicit failure accounting is necessary to prevent silent survivorship bias, especially for methods operating near hardware limits.

Estimators, confidence intervals, and frontier reconstruction. Given $\mathcal{L}_{m,\theta}$, we compute plug-in estimators

$$\widehat{\mu}_{m,\theta} = \frac{1}{n}\sum_{t=1}^{n} u_{m,\theta,t}, \qquad \widehat{\nu}_{m,\theta} = \frac{1}{n}\sum_{t=1}^{n} \mathbf{c}_{m,\theta,t},$$

together with confidence intervals for each scalar coordinate using empirical Bernstein (or a sub-Gaussian bound when mandated by the track). We then form an estimated achievable set $\widehat{\mathcal{A}}_m = \{(\widehat{\mu}_{m,\theta}, \widehat{\nu}_{m,\theta}) : \theta \in \Theta_m\}$ and compute its Pareto-efficient subset by removing empirically dominated points (optionally with a dominance certification rule based on one-sided bounds when the track emphasizes conservative dominance). Frontier functionals are computed on the resulting set: for a budget B, $\widehat{\mathrm{Acc}@B}$ is obtained by maximizing $\widehat{\mu}_{m,\theta}$ over feasible θ under $\widehat{\nu}_{m,\theta} \leq B$ (or $\widehat{\nu}_{m,\theta} \leq \mathbf{B}$ coordinate-wise), and $\widehat{\mathrm{AUF}}$ is computed by numerical integration over a published grid of budgets. Hypervolume is computed relative to a published reference point with a deterministic tie-handling rule (e.g., lexicographic ordering in the cost coordinate) to ensure reproducibility. When the track requests uncertainty for frontier functionals, we propagate uncertainty either via a union bound over knob settings (conservative) or via bootstrap over episodes with fixed knob grid (exploratory), with the chosen procedure fixed by the benchmark specification.

Warmup, caching, and compilation without bias. Accurate cost measurement requires careful treatment of system-level effects. We therefore distinguish *initialization* from *per-episode execution*. Initialization includes one-time actions such as loading weights into memory, building retrieval indices declared as part of the method, or compiling kernels. The track specifies whether these are included in reported costs; by default, we report per-episode costs and separately report one-time initialization costs so that deployment scenarios can be reconstructed. To reduce variance from cold-start effects without biasing comparisons, the harness executes a fixed warmup procedure prior to the first measured episode for each method (and, if necessary, for each θ), and excludes warmup from logged per-episode costs. Warmup length and content are fixed by protocol (e.g., a single synthetic episode with representative sequence lengths), preventing methods from choosing favorable warmup workloads. Caching is permitted only when it is (i) method-internal, (ii) does not depend on query labels, and (iii) does not exploit cross-episode leakage. For example, caching a tokenized prompt template is permitted; caching episode-specific model outputs for reuse on future episodes is not. If a method uses retrieval, any caching of retrieval results must be keyed by the declared input and must be accounted for in costs; moreover, the retrieval corpus must be declared under the track rules.

Batching and throughput normalization. Because batching can change both throughput and energy, we require a consistent batching policy. By default, we disallow batching across *different* episodes (to avoid subtle dataflow interactions and to simplify accounting), but permit batching within an episode (e.g., evaluating multiple query examples in mini-batches). If a method exposes batch size as a knob, it must be declared as part of θ and evaluated accordingly, and all costs (including padding waste and memory-bound slowdowns) are attributed to that choice. The harness records wall-clock time in addition to C^{J} to enable secondary analyses (e.g., energy-delay products) while keeping leaderboard definitions cost-coordinate driven.

These rules jointly yield an evaluation procedure that is (i) statistically interpretable through i.i.d. episode sampling and paired comparisons, (ii) reproducible through deterministic logging and fixed protocols, and (iii) cost-faithful through standardized instrumentation, warmup, and caching constraints.

Theory: what can and cannot be inferred from n episodes. We formalize the harness as drawing i.i.d. episodes $e_1, \dots, e_n \sim \mathcal{Q}$ and returning, for each method m (and knob setting $\theta \in \Theta_m$ when applicable), samples $\{(u_{m,\theta,t}, \mathbf{c}_{m,\theta,t})\}_{t=1}^{n}$ of the random pair $(U_{m,\theta}(e), \mathbf{C}_{m,\theta}(e))$. The central theoretical question is then: given bounded (or sub-Gaussian) utility and cost coordinates, how many episodes are required to estimate (i) mean utility and mean costs, (ii) frontier-based functionals such as Acc@B, AUF, and hypervolume, and (iii) method rankings, all with explicit (ε, δ) guarantees. The answer, under our assumptions, is that the evaluation is statistically well-posed with sample complexity $\Theta((\sigma^2/\varepsilon^2)\log(1/\delta))$, and that this dependence is information-theoretically unavoidable.

Tight sample complexity for mean utility and mean costs. Fix a method m and a knob θ. Under boundedness $U_{m,\theta}(e) \in [0,1]$ and each scalar cost coordinate $C_{m,\theta,k}(e) \in [0, C_{\max,k}]$, the empirical means

$$\widehat{\mu}_{m,\theta} = \frac{1}{n}\sum_{t=1}^{n} u_{m,\theta,t}, \qquad \widehat{\nu}_{m,\theta,k} = \frac{1}{n}\sum_{t=1}^{n} c_{m,\theta,t,k}$$

concentrate at rate $O(\sqrt{(\log(1/\delta))/n})$ by Hoeffding. When episode-to-episode variance is significantly smaller than the worst-case bound, empirical Bernstein inequalities yield tighter, data-adaptive confidence radii of the form

$$|\widehat{\mu}_{m,\theta} - \mathbb{E}[U_{m,\theta}]| \;\leq\; O\left(\sqrt{\frac{\widehat{\mathrm{Var}}(U_{m,\theta})\log(1/\delta)}{n}} + \frac{\log(1/\delta)}{n}\right)$$

(and analogously for each cost coordinate), which is the regime typically encountered once tasks are fixed and only episodes vary. Conversely, minimax lower bounds via two-point (Le Cam) constructions show that no estimator can guarantee ε-accurate mean estimation with probability $1-\delta$ using $o((1/\varepsilon^2)\log(1/\delta))$ episodes in the worst case. Thus, in the absence of additional structure on $\mathcal{Q}$, the harness cannot be made substantially more sample-efficient by clever post-processing; improvements come primarily from variance reduction (paired evaluation, stratification) rather than asymptotic rate changes.

Frontier functionals: stability of $\mathrm{Acc@}B$, AUF, and hypervolume. Cost-aware reporting is inherently a two-stage procedure: first estimate the achievable set $\mathcal{A}_m = \{(\mu_{m,\theta}, \nu_{m,\theta}) : \theta \in \Theta_m\}$ (or its Pareto frontier), then apply a functional ϕ (maximum under a budget, integral over budgets, or dominated hypervolume). The estimation error in $\widehat{\phi}_m = \phi(\widehat{\mathcal{A}}_m)$ decomposes into (i) statistical error from $(u_t, \mathbf{c}_t)$ sampling, and (ii) discretization/coverage error from the knob grid Θ_m. Under a Lipschitz condition on ϕ with respect to perturbations of points in $\mathbb{R}^{1+d_C}$, the plug-in estimator inherits the episode-level concentration: if the mean estimates for all knob settings are uniformly within ε (in the appropriate norm), then $|\widehat{\phi}_m - \phi(\mathcal{A}_m)| \le O(L_\phi \varepsilon)$. Uniformity over $\theta \in \Theta_m$ is obtained by a union bound (or, more sharply, by controlling the maximum deviation in a finite class), giving episode complexity scaling like

$$n \;=\; O\!\left(\frac{\sigma^2}{\varepsilon^2}\log\frac{|\Theta_m|}{\delta}\right)$$

for a fixed discrete sweep. This term makes explicit a practical design principle: the benchmark should prefer *coarse but meaningful* knob grids (e.g., logarithmic ranks/steps) over excessively dense sweeps that inflate $\log|\Theta_m|$ and hence required episodes for the same confidence.

Ranking stability: paired evaluation and episode-wise differences. Rankings are functions of $\{\mathbb{E}[U_m]\}_{m\in\mathcal{M}}$ (or of scalarized frontier metrics), and their stability depends on how quickly we can resolve gaps. For two methods m_1, m_2, the paired design induces episode-wise differences $D_t = U_{m_1}(e_t) - U_{m_2}(e_t)$. Concentration of $\bar{D} = \frac{1}{n}\sum_t D_t$ depends on $\tau^2 = \mathrm{Var}(D_t)$, not on $\mathrm{Var}(U_{m_1}) + \mathrm{Var}(U_{m_2})$. When methods succeed and fail on similar episodes (a common phenomenon), $U_{m_1}(e)$ and $U_{m_2}(e)$ are positively correlated and τ^2 is reduced, improving the number of episodes needed to assert $\mathbb{E}[U_{m_1}] > \mathbb{E}[U_{m_2}]$ by margin Δ:

$$n \;=\; \Theta\!\left(\frac{\tau^2}{\Delta^2}\log\frac{1}{\delta}\right).$$

This explains why paired episode sequences are not only a fairness constraint but also a statistical efficiency device: without pairing, one effectively replaces τ^2 by an unpaired variance that can be substantially larger, producing less stable leaderboards for a fixed evaluation budget.

Minimax lower bounds for frontier-aware leaderboards. Lower bounds extend beyond mean estimation. Any method-agnostic procedure that outputs a scalar score $\widehat{S}_m$ meant to approximate a frontier functional $S_m = \phi(\mathcal{F}_m)$ must contend with worst-case pairs of task distributions $\mathcal{Q}_0, \mathcal{Q}_1$ that (i) induce nearly indistinguishable per-episode observations for n samples, yet (ii) differ in S_m by $\Omega(\varepsilon)$. Packing arguments (of Fano type) yield that $n = \Omega((\sigma^2/\varepsilon^2)\log(1/\delta))$ episodes are required, up to constants, to guarantee $|\widehat{S}_m - S_m| \leq \varepsilon$ with probability $1 - \delta$. Informally, frontier metrics cannot be estimated substantially faster than the underlying means from which they are computed, unless one imposes additional structure on how utility varies with costs across θ.

Hardness of portfolio optimization under budgets. The benchmark primarily reports per-method frontiers; nevertheless, it is natural to ask for a *portfolio* that chooses among configurations (m, θ) under a global budget. This selection problem is computationally hard even when episode utilities and costs are known exactly. In the simplest variant, suppose we have a finite set of candidate configurations $i \in \{1, \dots, N\}$, each with value v_i (expected utility gain) and cost w_i (expected energy or FLOPs), and we seek $\max \sum_i v_i x_i$ subject to $\sum_i w_i x_i \leq B$, $x_i \in \{0, 1\}$. This is precisely 0-1 knapsack, implying NP-hardness of globally optimal budgeted selection. A second natural variant arises when choosing a limited set of prompts/demonstrations/retrieval exemplars intended to cover diverse episodes under a context-length budget; this can encode maximum coverage (and hence set cover hardness). These reductions justify our design choice: the benchmark reports frontiers and frontier functionals as primary artifacts, and treats any portfolio track (if included) as requiring explicit approximation or heuristic commitments rather than claiming global optimality.

Approximation options and what guarantees remain. Although exact portfolio optimization is NP-hard, meaningful approximation guarantees are available under additional assumptions. If portfolio utility is monotone submodular (diminishing returns) and costs are additive, the standard greedy algorithm achieves a $(1 - 1/e)$-approximation under a knapsack constraint. However, the benchmark cannot assume submodularity in general: adaptation configurations can interact in non-diminishing ways (e.g., complementary prompts), and costs may be non-additive due to caching or shared initialization. Therefore, whenever approximation algorithms are used in an auxiliary track, we require (i) a precise statement of the assumed utility model, (ii) explicit accounting for shared costs, and (iii) reporting of achieved solutions alongside the per-method frontiers so that the portfolio result is interpretable as a *policy layer* atop the measured primitives rather than as a replacement for them.

9 Experimental Protocol and Reference Baselines

Our experimental protocol is implementation-facing: it specifies (i) the required method interface, (ii) the episode- and hardware-level measurement rules that instantiate $\mathbf{C}_m$, (iii) the knob grids Θ_m for constructing $\widehat{\mathcal{F}}_m$, and (iv) a set of reference baselines that collectively span classical metric-based meta-learning, gradient-based meta-learning, and modern transfer/ICL adaptation.

Standardized method API and episode handling. Each submission exposes two entry points:

$$\mathsf{state} \leftarrow \mathsf{adapt}(S; \theta, \mathsf{seed}), \qquad \widehat{y}_Q \leftarrow \mathsf{evaluate}(Q, \mathsf{state}; \theta, \mathsf{seed}).$$

The episode object $e = (S, Q, \text{meta})$ is immutable and identically provided across methods (paired design). We require that adapt may read S (including any unlabeled portion when the track allows it) but may not access Q labels. Any stochasticity (dropout, sampling, augmentation) must be driven by the provided seed and logged. For methods with retrieval, the retriever is part of the method; any corpus beyond the benchmark-provided resources must be declared and evaluated only in the appropriate track.

Cost measurement protocol (HW-pinned). Costs are measured under a fixed HW profile (device type, driver/CUDA versions, mixed-precision settings, and power sampling toolchain). For each (m, θ, e_t), we record:

- $C^{\text{tok}}_{m,\theta}(e_t)$: prompt/context tokens consumed in *all* model calls, including calibration prompts and any verification/self-consistency calls;
- $C^{\text{flop}}_{m,\theta}(e_t)$: FLOPs estimate for adaptation plus inference, computed from logged tensor shapes and kernel-level proxies (or a declared analytic model), and reported with the estimator type;
- $C^{\text{param}}_{m,\theta}(e_t)$: effective parameters updated (e.g., LoRA matrices only, bias-only for BitFit, or full model for finetuning);
- $C^{\text{J}}_{m,\theta}(e_t)$: energy in Joules, measured as the time integral of device power over the adapt+evaluate region, with a standardized warmup and synchronization barrier;
- $C^{\text{CO}_2}_{m,\theta}(e_t)$: a derived proxy using declared carbon intensity and PUE assumptions attached to HW;
- $C^{\text{comm}}_{m,\theta}(e_t)$: bytes/rounds for federated variants (if enabled), including optimizer state when transmitted.

We fix batching rules (no cross-episode batching unless explicitly allowed by the track) to prevent amortizing costs in a way that invalidates per-episode accounting. When energy measurements are noisy, we permit r repeated executions for a small subset of episodes for variance estimation; the repeats must be performed for all methods on the same subset.

Reference baseline suite. We include baselines intended to be strong but transparent, with minimal tuning beyond Θ_m sweeps.

1. **Metric-based few-shot (no gradient adaptation).** *SimpleShot* (nearest class centroid with cosine distance, optionally with feature normalization) and *ProtoNets* (episodic prototype classifier). For these, the main knob is the backbone choice (fixed across baselines within a track) and optional transductive normalization when unlabeled support is permitted.

2. **Gradient-based meta-learning.** *MAML* and *First-Order MAML*, with θ including inner-loop steps $s \in \{1, 5, 10\}$, inner learning rate schedule, and whether we adapt all layers or only the head (*ANIL*-style). We treat meta-training compute as out of scope for $\mathbf{C}_m$ (benchmark-time cost measures adaptation+inference); nevertheless, we require reporting meta-training resources separately for completeness.

3. **Transfer and finetuning.** Full finetuning of a pretrained encoder with a linear head, and PEFT variants: *LoRA* (rank r), *adapters* (bottleneck size), *IA*^{3} (multiplicative vectors), and *BitFit* (bias-only). Here θ includes steps s, learning rate, and the PEFT capacity parameter (e.g., $r \in \{2, 4, 8, 16\}$).

4. **In-context learning (ICL) and calibrated ICL.** A prompt-based baseline that formats S as demonstrations and queries Q in a standardized template; θ includes prompt length budget L (demonstration count), exemplar selection rule (fixed heuristic such as random or class-balanced), and decoding configuration. We include a calibrated ICL variant that applies either contextual calibration (e.g., label prior correction via content-free prompts) or temperature scaling fit on S (when labels exist), counting all additional model calls in C^{tok} and C^{J}.

This list is not meant to be exhaustive; rather, it spans qualitatively different adaptation mechanisms so that frontier comparisons are meaningful.

Required ablations: isolating confounders. To reduce the degrees of freedom that otherwise invalidate comparisons, we require the following ablations when a method family is claimed to improve the frontier.

- **Budget sweeps.** Each method must report a sweep over its dominant knob(s): for ICL, L (and, when applicable, number of model calls); for PEFT/finetuning, steps s and capacity (e.g., LoRA rank r); for meta-learners, inner steps s. At minimum, we require a logarithmic grid (e.g., $s \in \{1, 2, 5, 10\}$) to support a nontrivial $\widehat{\mathcal{F}}_m$.

- **Prompt ordering and formatting.** For prompt-based methods, we evaluate both a canonical ordering (class-balanced, fixed) and a permuted ordering; we report the induced change in $\widehat{\mu}_{m,\theta}$ and in frontier metrics, since ordering effects can dominate small gains.

- **Domain shift strata.** Episodes are labeled by meta into shift regimes (in-domain, cross-domain, cross-label-space where applicable). We require stratified reporting so that a method that improves only a narrow slice cannot be presented as globally superior.

- **Calibration and thresholding.** When utility depends on thresholds (e.g., F1), we require either a fixed global threshold or a support-fit rule that is identical across methods; any additional fitting must be costed.

What constitutes strengthening evidence. We regard claims as stronger when they survive the following checks, all of which are directly supported by the logged $\mathcal{L}_m$.

1. **Frontier dominance with uncertainty.** Rather than reporting a single operating point, we present $\widehat{\mathcal{F}}_m$ with confidence bands induced by per-episode uncertainty, and we prefer statements of the form "method m_1 dominates m_2 on $[B_{\min}, B_{\max}]$" where dominance is robust to confidence intervals.

2. **Paired comparisons.** For any headline improvement, we report paired episode-wise differences in the relevant scalar metric (utility or $\phi(\widehat{\mathcal{F}}_m)$), along with an explicit (ε, δ) claim or a confidence interval for the gap.

3. **Sensitivity analysis.** We vary HW within a small approved set (when feasible) or, minimally, we report both token/FLOP costs and energy so that conclusions do not hinge on a single measurement modality.

4. **Ablation completeness.** When a method includes multiple components (e.g., retrieval plus calibration, or PEFT plus test-time ensembling), we require a component drop-out study where the cost increments are also reported; otherwise, improvements are not attributable.

Under this protocol, the empirical artifacts (logs, knob sweeps, and $\widehat{\mathcal{F}}_m$) are sufficient for reproducing the leaderboard functionals and for diagnosing whether an apparent gain is a genuine shift of the utility–cost tradeoff or merely movement along it.

10 Discussion

CarbonBench-FSL alters the qualitative conclusions one draws from few-shot results by elevating *utility–cost tradeoffs* to the primary object of comparison. In a classical accuracy-only view, it is common to treat meta-learning, transfer/finetuning, and in-context learning (ICL) as competing families and to report a single best-tuned point per family. In our formulation, each method m induces an achievable set of random outcomes $(U_m(e), \mathbf{C}_m(e))$ across episodes $e \sim \mathcal{Q}$, and the benchmark output is an estimate of $\mathcal{F}_m$ (with uncertainty) rather than a single operating point. This shift is not cosmetic: it changes what it means for one family to "win." A method that is best at a high-compute operating point may be strictly dominated at deployment-relevant budgets, and conversely a method with modest peak accuracy may be Pareto-optimal in low-energy or low-token regimes.

Meta-learning versus transfer under adaptation-time costs. When we restrict $\mathbf{C}_m$ to benchmark-time adaptation and inference, gradient-based meta-learners frequently occupy an intermediate region of the frontier: they can achieve nontrivial improvements with small inner-loop step counts, but their marginal utility per additional inner step may saturate quickly once the support set S is small. In contrast, transfer with PEFT (e.g., low-rank updates) tends to offer a wider knob range: by varying effective capacity C_m^{param} and steps (and hence C_m^{flop} and C_m^{J}), one often traces a longer, smoother frontier. The cost-aware view therefore encourages a more precise statement: meta-learning may be preferable when a strict bound on per-episode state update is imposed (e.g., small C_m^{param} and bounded steps), whereas PEFT may dominate when modest additional energy is acceptable and the task distribution $\mathcal{Q}$ exhibits sufficient transfer for gradient updates to be effective. Importantly, this is compatible with the fact that meta-training may be expensive; CarbonBench-FSL separates *benchmark-time* $\mathbf{C}_m$ from *precomputation* resources, so that conclusions about deployment-time budgets are not confounded by training-time sunk costs, while still allowing those costs to be reported for completeness.

ICL versus parameter updates under token and energy budgets. ICL changes the cost accounting primarily through $C_m^{\mathrm{tok}}(e)$ and the number of model calls, which then map to $C_m^{\mathrm{J}}(e)$ and $C_m^{\mathrm{CO_2}}(e)$ under HW. On many contemporary systems, energy and latency scale more predictably with tokens than with nominal FLOP counts, particularly when memory bandwidth or kernel launch overhead dominates. For this reason, an ICL method that appears “training-free” can nevertheless be expensive in C^{J} if it relies on long contexts, multiple decoding passes, or calibration/self-consistency. Conversely, small-step PEFT can be cost-competitive when the adaptation compute is limited and amortized over a short query set Q. CarbonBench-FSL thus changes the natural comparison: rather than asking whether ICL or finetuning achieves higher U at its best setting, we ask for which budget vectors B (tokens, energy, or combined) the corresponding frontiers intersect, and whether dominance persists across shift strata in meta. In particular, we find it more informative to report statements of the form

$$\forall B \in [B_{\min}, B_{\max}] : \quad \phi(\mathcal{F}_{m_1}; B) \geq \phi(\mathcal{F}_{m_2}; B),$$

with confidence bands, than to report a single “best” score.

Consequences for claims in the literature. The frontier-centric perspective weakens several common but under-specified claims. First, an improvement that moves a method along its own frontier (e.g., by spending more tokens or more gradient steps) is not evidence of a better adaptation mechanism; it is merely a choice of a different operating point. Second, small accuracy gains become ambiguous without cost normalization: a gain of Δ in $\widehat{U}_m$ may be negligible if it requires an order-of-magnitude increase in C^{J}_m or C^{tok}_m. Third, conclusions can flip under domain shift: methods that exploit support labels aggressively may excel in-domain but lose their advantage cross-domain, where the support signal is less predictive; stratified frontiers clarify whether a method is robustly Pareto-improving or only locally so. Finally, paired evaluation (same e_t across methods) makes it practical to report uncertainty on *gaps* rather than on absolute performance, which materially stabilizes rankings when differences are small.

Recommendations for benchmark extensions: modalities and task structure. Several natural extensions preserve the episodic formalism while broadening its scope. (i) *Multimodal episodes*: define S, Q over paired text–image or audio–text inputs, with modality-specific cost coordinates (e.g., vision encoder FLOPs) included in $\mathbf{C}_m$. (ii) *Structured prediction*: utilities such as exact match, edit distance, or constrained decoding reward can be accommodated by redefining U_m; in these settings, decoding strategies become a first-class knob and must be reflected in C^{tok}_m and C^{J}_m. (iii) *Interactive/decision episodes*: for bandit or reinforcement-learning-like tasks, one can treat an episode as a short horizon with utility equal to episodic return and costs accumulated over interaction; the same concentration-based estimation framework applies provided the episode sampler is i.i.d.

Privacy and federated adaptation. Cost-aware evaluation is particularly well suited to privacy-preserving and federated variants, where communication and local compute are dominant constraints. A principled extension is a federated track in which each episode e includes partitioned support $S = \{S^{(i)}\}_i$ across clients, and a method m is permitted client/server message passing that incurs $C_m^{\mathrm{comm}}(e)$ and client-side $C_m^{\mathrm{J}}(e)$. Utilities U_m can incorporate privacy-aware penalties (e.g., a constraint on $(\varepsilon_{\mathrm{DP}}, \delta_{\mathrm{DP}})$ if differential privacy is enforced) by either filtering invalid methods or scalarizing privacy into the utility/cost specification. The central methodological point is that privacy should not be treated as an afterthought: it is a constraint that reshapes the feasible frontier, and thus must be reported in the same language as tokens, energy, and communication.

Continual and amortized adaptation. A further extension is to relax the independence of method state across episodes. In continual settings, a method may carry parameters or memory across a sequence $e_1, e_2, \ldots$, improving U_m on later episodes at the risk of drift or catastrophic forgetting. CarbonBench-FSL can accommodate this by redefining the unit of evaluation to be a *block* (a sequence sampled from a higher-level distribution) and by reporting both per-episode and amortized costs, e.g.,

$$\overline{\mathbf{C}}_m = \frac{1}{T}\sum_{t=1}^{T} \mathbf{C}_m(e_t; \mathsf{state}_{t-1}), \qquad \overline{U}_m = \frac{1}{T}\sum_{t=1}^{T} U_m(e_t; \mathsf{state}_{t-1}),$$

with explicit rules governing what information may persist. This makes explicit a tradeoff that is otherwise obscured: methods that spend extra compute early (high initial C^{J}) may be preferable if they reduce steady-state costs or improve robustness under shift.

In summary, CarbonBench-FSL encourages conclusions that are conditional on budgets and shift regimes, expressed in terms of (estimated) Pareto dominance and frontier functionals with uncertainty. This shifts the default claim from "method family A is better than B" to "A offers a better utility–cost tradeoff on the relevant region of $\mathbf{C}$-space under the stated assumptions," which is the appropriate level of specificity for deployment-facing evaluation.

Limitations and ethics: measurement dependence and carbon accounting. A cost-aware benchmark is only as interpretable as its measurement protocol, and we emphasize that our reported coordinates $\mathbf{C}_m$ are not intrinsic properties of an algorithm in isolation. In particular, the energy coordinate $C_m^{\mathrm{J}}(e)$ and its derived proxy $C_m^{\mathrm{CO_2}}(e)$ are functions of the joint algorithm–systems configuration: model implementation details, compiler and kernel choices, batch scheduling, memory pressure, and the hardware profile HW. Two methods m_1, m_2 can change their relative ordering in C^{J} when moved from one accelerator to another (e.g., due to different arithmetic intensity or KV-cache behavior), even if their token counts and nominal FLOP estimates are comparable. Consequently, any statement of the form "m_1 is more energy efficient than m_2" must be read as conditional on a stated

HW and a stated protocol (warmup, batching, precision, decoding settings). We therefore treat HW as part of the benchmark definition rather than an incidental experimental detail, and we regard portability across hardware as an empirical question to be evaluated by rerunning the protocol under an alternative HW'.

A second limitation concerns facility-level overhead, commonly summarized by power usage effectiveness (PUE). If one reports a facility-adjusted energy $C^{\mathrm{J}}_{\mathrm{fac}} = \mathrm{PUE} \cdot C^{\mathrm{J}}_{\mathrm{IT}}$, then the value of PUE is not only site dependent but also time varying and often not measurable by an external evaluator. Moreover, shared infrastructure (cooling, networking, storage) introduces allocations that can be accounted for in multiple defensible but non-equivalent ways. For this reason, when we compute a CO_2 proxy we do not claim to measure an objective "true carbon footprint" of an episode. Rather, we recommend that benchmark reports explicitly separate (i) a directly measured or instrumented IT-energy estimate $C^{\mathrm{J}}_{m}(e)$ under the protocol and (ii) a derived quantity $C^{\mathrm{CO}_2}_{m}(e) = \kappa \cdot C^{\mathrm{J}}_{m}(e)$, where κ denotes an explicitly stated carbon-intensity factor that may incorporate PUE assumptions. This factorization makes the normative and contextual components of carbon reporting explicit, and it allows downstream users to substitute their own κ to reflect a different region, grid mix, or accounting standard without rerunning the entire evaluation.

Risks of proxy misuse and misinterpretation. Because C^{CO_2} is necessarily a proxy, there is a nontrivial risk of misuse. First, a single scalar carbon number can be over-interpreted as an absolute environmental claim, masking the fact that real-world emissions depend on time-of-day, marginal grid intensity, and load shifting. Second, carbon proxies can be used to create an appearance of environmental rigor without reflecting meaningful deployment conditions (e.g., choosing a favorable κ while ignoring that the intended deployment region differs). Third, proxy reporting can incentivize narrow optimization on reported coordinates while ignoring unmeasured externalities, such as embodied emissions of hardware, amortization over utilization, or the impact of repeated hyperparameter search. While we support transparent reporting of those factors where feasible, we do not assume that CarbonBench-FSL can, by itself, enforce a complete life-cycle assessment. Our position is that the benchmark should avoid false precision: it should prefer decomposed reporting ($C^{\mathrm{tok}}, C^{\mathrm{flop}}, C^{\mathrm{J}}$ with uncertainty, plus an optional C^{CO_2} under declared assumptions) over a single aggregated claim that obscures the accounting choices.

Another misuse risk concerns cross-paper comparisons. If two studies use different measurement protocols, one may observe differences in C^{J} that are dominated by instrumentation, implementation, or batch-size effects rather than by algorithmic structure. In particular, token-based costing C^{tok} is often more stable across platforms than energy costing, yet it too can be misleading when methods differ in decoding strategy (beam search, self-consistency, tool calls) or in the number of model invocations. For these reasons, we view a benchmark run as a paired and internally consistent comparison under a fixed protocol, and we caution against comparing energy or carbon numbers across unrelated experimental setups without harmonizing HW, decoding parameters, and load conditions.

Responsible reporting guidelines. To reduce ambiguity and to make ethical use of cost reporting more likely, we recommend the following minimum disclosure for any CarbonBench-FSL result table or submission:

- *Protocol and hardware disclosure:* report HW (accelerator type, driver/runtime versions, precision settings), batching rules, warmup rules, and decoding configuration; report whether power was measured (e.g., via onboard sensors) or inferred.
- *Primary cost coordinates:* report C^{tok} and C^{J} (with uncertainty bands or repeated-run variance estimates when available) in addition to any derived $C^{\mathrm{CO_2}}$; do not report $C^{\mathrm{CO_2}}$ without the underlying C^{J}.
- *Carbon proxy assumptions:* if $C^{\mathrm{CO_2}}$ is reported, state the exact multiplier κ used (including any PUE factor) and its provenance; if κ varies across runs, state the averaging scheme.
- *Separation of benchmark-time and precomputation:* clearly separate adaptation/inference costs (the benchmark target) from any offline costs such as meta-training, supervised pretraining, index construction for retrieval, or hyperparameter sweeps; if offline costs are large, report them as supplementary totals rather than silently amortizing them.
- *Budget-conditional claims:* express conclusions as comparisons of frontiers or of budgeted functionals (e.g., accuracy@B, AUF) rather than absolute statements detached from B; avoid implying dominance outside the measured budget region.

These guidelines are not merely stylistic: they are required for the mathematical object we estimate—a frontier conditional on HW and on a protocol—to be well defined and reproducible.

Ethical scope and non-goals. We also delimit what the benchmark does not address. First, while cost-aware evaluation may discourage needlessly expensive adaptation procedures, it does not by itself ensure that a deployment is socially desirable; energy and carbon are only two axes among many. Second, CarbonBench-FSL does not adjudicate which accounting standard for emissions is correct; we expose assumptions so that the reader can reinstantiate them. Third, an algorithm may reduce measured energy while increasing other harms (e.g., by shifting compute to unreported external services, or by relying on proprietary data). Our fairness constraint therefore requires disclosure of external retrieval corpora and auxiliary models, but disclosure cannot guarantee completeness. Finally, privacy-sensitive settings introduce additional ethical constraints: even if a method is Pareto-optimal in $(U, \mathbf{C})$, it may be unacceptable if it violates privacy or safety requirements. We therefore encourage tracks in which such constraints are made explicit (e.g., by filtering or by adding a coordinate/constraint), rather than treating them as informal caveats.

In sum, we view measurement dependence as an inherent feature of systems-aware evaluation, not a defect to be ignored. The appropriate response is to specify HW and the protocol as part of the benchmark, to report decomposed costs with uncertainty, and to treat any carbon quantity as a conditional proxy under stated assumptions. Under these practices, cost-aware reporting can improve scientific clarity and reduce incentives for misleading accuracy-only optimization, while remaining honest about what is and is not measured.