Quality-Adjusted Intelligence per Joule

A boundary-explicit measure of useful AI task service

Matthew Long
The YonedaAI Collaboration
YonedaAI Research Collective
Chicago, IL
matthew@yonedaai.com · https://yonedaai.com

23 July 2026

Abstract

Tokens per joule is an engineering throughput ratio. It is not invariant to tokenizer choice or verbosity, and it assigns the same output unit to a correct solution and a plausible error. This paper defines quality-adjusted AI task service per joule for a declared task distribution, difficulty convention, quality rule, correctness grader, reliability design, usefulness weight, latency target, and energy boundary. The primary statistic is a ratio of expected service to expected energy. It keeps energy from failed and late attempts in the denominator. We prove that tokens per joule is not tokenizer invariant, construct token-volume and task-mix rank reversals, and define an energy-quality frontier that preserves tradeoffs hidden by a single ratio. A measurement protocol connects MLCommons wall-power practice with the scenario-and-metric discipline of HELM, the statistical guidance of NIST, and METR's task-duration approach to within-domain difficulty. A standard-library Python implementation supplies seeded bootstrap intervals, Wilson intervals, rank-reversal fixtures, and Pareto-frontier checks. The result is PROVED for the algebraic non-invariance statements and COMPUTATIONAL for the executable examples. It is not a measure of general intelligence or realized economic value.

Keywords: AI evaluation; energy efficiency; inference; benchmarking; task success; reliability; Pareto frontier; measurement

Contents

1 Introduction

An inference server can produce more tokens per joule by changing its tokenizer, increasing verbosity, returning repetitive text, or choosing easier requests. None of these changes establishes that the server performs more useful work. Token throughput remains valuable for capacity planning because providers schedule memory, bandwidth, and decoding around tokens. The mistake is to promote that internal accounting unit into an output measure of machine capability.

The measurement problem has two layers. First, an evaluator must specify the service being measured. A coding agent, translation system, retrieval model, and medical classifier do not emit a common natural unit. Second, the evaluator must specify the energy boundary. Accelerator telemetry, host energy, measured wall energy, facility energy, and lifecycle energy answer different questions. A ratio assembled from an unspecified numerator and an unspecified denominator cannot support a comparison.

We use the phrase "quality-adjusted intelligence per joule" because it names the research program, but the operational object in this paper is narrower: *quality-adjusted AI task service per joule*. The qualification matters. The metric concerns performance on a declared task population. It does not estimate a latent, domain-independent quantity called general intelligence. It also stops before realized economic value. The latter requires a causal chain from an output to adoption, displacement or complementarity, downstream effects, and a counterfactual value functional.

The paper makes five contributions.

1. It defines a reproducible measurement tuple for energy-normalized AI task service and makes task distribution, difficulty, correctness, reliability, usefulness, latency, and energy scope explicit.
2. It proves that tokens per joule is not invariant to tokenization and gives a strict ranking reversal between token throughput and verified task service.
3. It shows that quality-adjusted rankings remain distribution-specific. A change in task mix can reverse two system rankings even when each task-level score is held fixed.
4. It defines an energy-quality frontier under a latency constraint. The frontier is preferred to a scalar ratio when users have different energy budgets or minimum quality requirements.
5. It supplies executable reference code for the estimator, percentile bootstrap, Wilson reliability intervals, rank reversals, and Pareto filtering.

The paper is deliberately conservative. Difficulty is not inserted as a universal multiplier. The primary report is stratified by a declared difficulty measure, and a difficulty-weighted scalar appears only as a sensitivity analysis. Likewise, "usefulness" means a benchmark service weight fixed before evaluation. It is not revenue, profit, welfare, or labor displacement.

1.1 Evidence status

The program uses explicit status labels:

- PROVED denotes a complete argument under stated assumptions;
- CONDITIONAL denotes a result that depends on a measurement or identification design;
- COMPUTATIONAL denotes executable evidence;

- OBSTRUCTED denotes a comparison blocked by incompatible data or assumptions;
- OPEN denotes an unresolved question.

The tokenizer non-invariance result and the algebra of the estimator are PROVED. The rank-reversal fixtures, bootstrap implementation, and frontier algorithm are COMPUTATIONAL. Whether a chosen benchmark generalizes to a deployment population is CONDITIONAL. A universal conversion from benchmark points to economic value is OBSTRUCTED without a separate causal design.

2 What existing evaluation practice already provides

2.1 Energy measurement

MLCommons has developed power-measurement procedures alongside MLPerf Inference. Its documentation describes full-system alternating-current measurement with approved analyzers and couples the power interval to the benchmark run [2, 1]. This is a strong operational starting point. It avoids treating thermal design power as measured energy and captures more of the system than accelerator telemetry alone.

The scope remains benchmark-specific. A wall measurement of the system under test does not automatically include upstream networking, cooling beyond the metered boundary, embodied hardware, or data-center construction. Conversely, multiplying by a power usage effectiveness value changes the boundary and adds uncertainty. An evaluator should not label a measured operational result "lifecycle energy" unless a non-overlapping lifecycle inventory has actually been constructed.

Prior work on Green AI called for reporting computational cost alongside accuracy [8]. Henderson and colleagues proposed systematic energy and carbon reporting [9]. These efforts establish that resource disclosure is part of responsible empirical work. They do not settle the output unit problem for generative or agentic systems.

2.2 Quality and scenario coverage

HELM argues that model evaluation should declare scenarios, adaptations, and metrics rather than infer broad capability from one aggregate score [3]. That structure is useful here. Energy is measured for a particular system under a particular evaluation protocol, while quality is defined over the same requests and service conditions. Neither half should be borrowed from an unmatched leaderboard.

NIST's work on AI test, evaluation, validation, and verification emphasizes construct validity, reliability, context, and uncertainty [4]. NIST AI 800-3 distinguishes performance on a fixed benchmark from generalized performance over a broader population of similar questions [5]. That distinction applies directly to an energy-normalized ratio. A narrow confidence interval around a benchmark ratio does not prove that the ratio transports to a different user population.

2.3 Difficulty and task duration

METR defines a task-completion time horizon using the duration that human experts require for tasks that an AI system completes with a specified probability [6, 7]. Human duration is attractive because it provides an interpretable within-domain ordering and captures some forms of task length. It is not a universal difficulty scale. Tool familiarity, domain expertise, environmental setup, task ambiguity, and the selected human population all affect the duration. METR's own methodology reports hierarchical uncertainty and limits claims to the studied task domain.

This paper therefore permits several declared difficulty conventions: expert-time bins, item-response estimates, rubric complexity, or no weighting. Comparisons must use the same convention. The unweighted and stratified results remain primary.

3 Measurement tuple and system boundary

Definition 3.1 (Evaluation tuple)**.** An energy-normalized AI task evaluation is indexed by

$$\mathcal{A} = (b, \mu, d, \delta, q, c, r, u, L, \tau, \rho, \omega, \pi),$$

where b is the energy boundary, μ is the target task distribution, d is a difficulty descriptor, δ is its optional normalized dimensionless weight, q is a quality rule, c is a correctness rule, r is a reliability design, u is a declared service weight, L is a latency or service-level constraint, τ is the measurement interval, ρ is the shared-infrastructure allocation rule, $\omega_x(\cdot \mid t)$ is the stochastic law of outputs and environment states for system x given task t, and π is the statistical uncertainty procedure.

The tuple prevents a common reporting failure. Two papers may both print "joules per answer" while one measures only GPU energy, the other measures wall energy, one excludes failed requests, and the other uses a different task mix. The printed units match, but the estimands do not.

3.1 Task population

Let $\mathcal{T}$ be a finite or measurable set of task instances and let μ be the target distribution. A benchmark average usually substitutes the empirical distribution

$$\widehat{\mu}(t_i) = \frac{1}{n}.$$

That substitution is defensible only if the benchmark itself is the target. If the target is a deployment population, the evaluator needs sampling or importance weights. For observed task i, let $w_i \geq 0$ denote its weight, with normalized weight

$$\widetilde{w}_i = \frac{w_i}{\sum_{j=1}^{n} w_j}.$$

The distribution covers more than prompts. It includes tool availability, context length, input modality, required output format, time budget, and interaction policy. A change in any of these may define a new task population.

3.2 Difficulty

For task t, let $d(t)$ be a declared nonnegative difficulty descriptor. It may carry units, such as expert seconds. If difficulty enters a scalar service product, a published normalization maps it to a dimensionless weight $\delta(t)$. Raw $d(t)$ is used for stratification or model fitting and never enters a dimensionless product directly. Three reporting modes are allowed.

1. In *stratified mode*, d assigns a bin and the evaluator reports a separate score for each bin.
2. In *model-based mode*, d is estimated using a declared item response or regression model. The estimate and its uncertainty are reported.
3. In *weighted mode*, d is mapped to dimensionless normative weight δ by a published normalization. The unweighted result must appear beside it.

Human expert duration may support the first two modes within a task family. A ten-minute legal research task and a ten-minute image-editing task do not become interchangeable merely because their durations match. Cross-domain weighting needs an additional service model. Rasch models provide one established approach to within-domain item scaling, but their latent scale still depends on model assumptions and identifying constraints [12].

3.3 Quality and correctness

Quality $q_i \in [0, 1]$ records graded fitness for the task specification. Correctness $c_i \in [0, 1]$ records factual, logical, or test-based correctness. They may coincide for a binary exact-answer task. They may differ for a translation or program patch, where a response can pass a narrow test while violating style, safety, or maintainability requirements.

The grading rule must be fixed before system outputs are inspected. Automated graders require validation against human or executable ground truth. Model-based judges should report judge identity, prompt, order effects, position randomization, and agreement. If correctness already includes every quality dimension, set $q_i = 1$ rather than count the same property twice.

3.4 Reliability

Reliability $r_i \in [0, 1]$ is the probability that the system reproduces the required outcome across the declared variation in seeds, retries, transient failures, and environment. It is not a decorative confidence score. The sampling unit must match the intended claim.

For a binary task with x successes in n independent trials, the point estimate is $\widehat{r} = x/n$. A Wilson interval is preferable to a naive normal interval near zero or one. If attempts are clustered by task family, model version, or site, a hierarchical model or cluster bootstrap is needed.

Correctness and reliability can overlap. If c_i is the mean success across all retries, multiplying by a separately estimated r_i would count failure twice. In that design set $r_i = 1$, or redefine c_i as conditional quality given a completed response. The manifest must say which design is used.

3.5 Usefulness without economic value

Usefulness $u_i \in [0, 1]$ is a declared service weight inside the evaluation. It can downweight tasks that are technically valid but irrelevant to the intended workload. It cannot be inferred from token count or benchmark difficulty.

This paper stops at verified service. Revenue, cost savings, welfare, and scientific value require an adopted action and a counterfactual downstream effect. Calling u_i an economic value coefficient would skip that causal work. Paper 5 in this program addresses the later chain.

3.6 Latency

Let $L^\star$ be the maximum acceptable latency. The simplest service indicator is

$$\ell_i(L^\star) = \mathbf{1}\{L_i \le L^\star\}.$$

Some applications may use a preregistered continuous penalty $\ell_i \in [0, 1]$. A penalty chosen after observing model results is a ranking control, not a neutral metric.

A late answer has zero or reduced service under the declared contract, but its energy remains in the denominator. The same rule applies to invalid outputs, refusals that violate the task contract, crashes, retries, and timeouts.

3.7 Energy boundary

We distinguish four profiles.

Device operational. Measured accelerator or device energy. Host, memory, networking, facility overhead, and embodied energy are excluded.

System operational. Energy for the complete system under test at the wall. Warm-up, idle allocation, batching, retries, and serving overhead are declared.

Facility operational. System energy plus measured or modeled facility overhead. Any PUE multiplier is indexed by site and interval.

Lifecycle. A non-overlapping allocation of operational and embodied components over a declared lifetime and utilization schedule.

Results from different profiles cannot be ranked without a bridge. Device telemetry may diagnose prefill and decode behavior, but it is not a substitute for wall energy in a deployed-system claim.

4 Quality-adjusted task service per joule

4.1 Task-level contribution

For attempt i, define the dimensionless service contribution

$$s_i = \delta_i u_i q_i c_i r_i \ell_i(L^\star). \tag{1}$$

Here δ_i is one when difficulty is only stratified. If a difficulty-weighted sensitivity result is reported, δ_i is the declared dimensionless normalization of raw descriptor d_i. Every factor is

Table 1: Minimum fields for a comparable AI energy result.

Field	Required declaration
Functional unit	One task attempt drawn from the named distribution
System	Model, weights, software, precision, hardware, serving stack
Task protocol	Prompt, tools, context, output rule, grader, retry policy
Energy scope	Device, system, facility, or lifecycle, with meter location
Shared load	Idle, warm-up, batching, memory, host, and allocation rule
Service rule	Difficulty, quality, correctness, reliability, usefulness
Latency	Deadline or continuous penalty fixed before evaluation
Uncertainty	Sampling unit, repetitions, interval method, meter error
Exclusions	Networking, cooling, embodied hardware, or other omitted scope

declared and nonnegative. A product is convenient because a zero in a mandatory dimension makes the contribution zero. It is not compulsory. An evaluator may replace the product with a preregistered aggregation function $g(\delta_i, u_i, q_i, c_i, r_i, \ell_i)$. The exact function is part of the estimand.

The product is most defensible when factors represent separate interfaces. For example, a completed answer can receive a correctness score, while reliability is estimated from independent reruns and usefulness is fixed by the target task distribution. If factors are empirically redundant, the evaluator should report a joint outcome instead of multiplying correlated grades.

Definition 4.1 (Quality-adjusted task service per joule)**.** For system configuration x, evaluation tuple $\mathcal{A}$, and positive expected operational energy, define

$$\mathrm{QAIJ}_{\mathcal{A}}(x) = \frac{\mathbb{E}_{\mu,\omega_x}[s(T,x)]}{\mathbb{E}_{\mu,\omega_x}[E_b(T,x)]}. \tag{2}$$

The unit is declared quality-adjusted task equivalents per joule.

The expectation in eq. (2) is over tasks drawn from μ and the system and environment data-generating law ω_x. The uncertainty procedure π, such as a hierarchical bootstrap or mixed model, estimates sampling uncertainty but is not itself the source of outcomes.

The ratio uses expected service divided by expected energy. It is not the expectation of per-task ratios:

$$\frac{\mathbb{E}[s]}{\mathbb{E}[E]} \neq \mathbb{E}\left[\frac{s}{E}\right] \quad \text{in general.}$$

The ratio of expectations answers a capacity question: how much expected service does the system provide for the expected energy spent on the target mix? The mean of ratios gives extreme influence to tasks with very small denominators and does not aggregate cleanly to fleet energy.

4.2 Finite-sample estimator

For weighted observations $i = 1, \ldots, n$,

$$\widehat{\mathrm{QAIJ}}_{\mathcal{A}}(x) = \frac{\sum_i w_i s_i}{\sum_i w_i E_{b,i}}. \tag{3}$$

Normalization by $\sum_i w_i$ cancels. Failed and late observations enter with $s_i = 0$ and their full measured energy $E_{b,i} > 0$.

Proposition 4.2 (Aggregation consistency)**.** *Suppose disjoint batches B_k use the same evaluation tuple. Let $S_k = \sum_{i \in B_k} w_i s_i$ and $E_k = \sum_{i \in B_k} w_i E_i$. The pooled estimate is the energy-weighted mean of batch estimates:*

$$\widehat{\mathrm{QAIJ}}_{\mathrm{pool}} = \sum_k \frac{E_k}{\sum_j E_j} \widehat{\mathrm{QAIJ}}_k.$$

Proof. Substitute $\widehat{\mathrm{QAIJ}}_k = S_k / E_k$:

$$\sum_k \frac{E_k}{\sum_j E_j} \frac{S_k}{E_k} = \frac{\sum_k S_k}{\sum_j E_j},$$

which is the pooled ratio in eq. (3). □

This property fails for an unweighted mean of task-level ratios. The pooled form also makes denominator auditing easier because reported service and energy totals remain separately inspectable.

4.3 Primary and sensitivity reports

A serious report should not print one QAIJ number. We recommend:

1. unweighted task service per joule;
2. service per joule by difficulty stratum;
3. a difficulty-weighted sensitivity result;
4. expected service and expected energy separately;
5. latency and reliability summaries;
6. the energy-quality frontier;
7. uncertainty intervals under alternative task mixes.

The vector makes normative choices visible. A procurement team may require a high reliability floor, while a batch research workflow may accept more variance for lower energy. A single weighted average conceals that choice.

5 Why tokens per joule is not an outcome metric

5.1 Tokenizer non-invariance

Let a tokenizer κ map the same output string y to a sequence of tokens. Define

$$\mathrm{TPJ}_{\kappa}(x) = \frac{\mathbb{E}[|\kappa(Y_x)|]}{\mathbb{E}[E_b(x)]}.$$

Theorem 5.1 (Tokenizer non-invariance)**.** *If two admissible tokenizers κ_1 and κ_2 assign different token counts to an output with positive probability, and the system output, task outcome, and measured energy are unchanged, then tokens per joule changes while* $\mathrm{QAIJ}_{\mathcal{A}}$ *remains unchanged.*

Proof. The denominators are identical by assumption. The token-count expectations differ because $\mathbb{P}(|\kappa_1(Y)| \neq |\kappa_2(Y)|) > 0$ and the expected counts are assumed unequal. Thus $\mathrm{TPJ}_{\kappa_1} \neq \mathrm{TPJ}_{\kappa_2}$. Equation (2) contains task service and energy, neither of which changes under a resegmentation of the same string. Therefore $\mathrm{QAIJ}_{\mathcal{A}}$ is unchanged. □

Remark 5.2. The theorem does not make token throughput useless. It shows that token throughput is conditional on a tokenizer and belongs to internal production accounting. A provider comparing two serving stacks for the same model and tokenizer may use it responsibly.

5.2 Verbosity rank reversal

Proposition 5.3 (Token-volume rank reversal)**.** *There exist systems A and B with equal energy such that A has higher tokens per joule and B has higher quality-adjusted task service per joule.*

Proof. Let each system consume 10 joules. System A produces 100 tokens and has service contribution 0.1. System B produces 10 tokens and has service contribution 1. Then

$$\mathrm{TPJ}(A) = 10 > 1 = \mathrm{TPJ}(B),$$

while

$$\mathrm{QAIJ}(A) = 0.01 < 0.1 = \mathrm{QAIJ}(B).$$

The constructed ranking reversal is strict. □

The reference tests instantiate this example under the names `wordy-wrong` and `concise-correct`. The test is COMPUTATIONAL evidence that the implementation matches the proposition. The proposition itself is PROVED.

5.3 Output length can still matter

Output length affects latency, memory traffic, serving capacity, user reading time, and energy. It therefore belongs in the causal path. The correct interpretation is not "tokens never matter." It is that token count is an intermediate quantity:

$$E \longrightarrow \text{compute and memory traffic} \longrightarrow \text{tokens} \longrightarrow \text{graded task outcome}.$$

A shorter correct answer may reduce both energy and user burden. A longer proof may be required for verifiability. The task rubric, not token volume by itself, decides which output is better.

6 Task distribution and ranking reversals

6.1 Distribution-specific efficiency

Consider two task classes, easy e and hard h, each costing 10 joules per attempt. System A has service scores

$$s_A(e) = 1, \qquad s_A(h) = 0.1,$$

while system B has

$$s_B(e) = 0.4, \qquad s_B(h) = 1.$$

Under an easy-heavy distribution $\mu_1(e) = 0.9$, $\mu_1(h) = 0.1$,

$$\mathrm{QAIJ}_{\mu_1}(A) = 0.091, \qquad \mathrm{QAIJ}_{\mu_1}(B) = 0.046.$$

Under a hard-heavy distribution $\mu_2(e) = 0.1$, $\mu_2(h) = 0.9$,

$$\mathrm{QAIJ}_{\mu_2}(A) = 0.019, \qquad \mathrm{QAIJ}_{\mu_2}(B) = 0.094.$$

The preferred system reverses.

Proposition 6.1 (Task-mix rank reversal)**.** *If two systems have crossing task-level service advantages and positive energy, then there exist task distributions that reverse their* QAIJ *ranking.*

Proof. Suppose A is more efficient on task e and B is more efficient on task h. A distribution concentrated sufficiently near e ranks A first by continuity of the finite mixture. A distribution concentrated sufficiently near h ranks B first. The explicit construction above provides one witness. □

This is not a defect in the metric. It reveals that "best model" is incomplete without a workload. Benchmark maintainers should publish the task weights, and deployers should rerun the estimator under their own target mix.

6.2 Difficulty shifts and Simpson effects

An aggregate can improve while every difficulty stratum worsens if task mix changes toward easier cases. Conversely, a system can improve within each stratum while its aggregate declines because the deployment admits harder work. Both are versions of a composition effect.

Every longitudinal report should therefore include:

- the marginal task distribution at each time;
- within-stratum quality and energy;
- a standardized estimate using a fixed reference distribution;
- the observed deployment estimate using the actual distribution.

The standardized estimate supports technical comparison. The deployment estimate supports operational accounting. They answer different questions.

6.3 Importance weighting

If benchmark tasks are sampled from proposal distribution ν but the target is μ, importance weights are

$$w(t) = \frac{\mu(t)}{\nu(t)}$$

where $\nu(t) > 0$ whenever $\mu(t) > 0$. The weighted ratio estimator uses eq. (3). Large weights increase variance and expose weak support. Trimming weights changes the estimand and must be reported.

If the benchmark contains no examples of a deployment task family, weighting cannot manufacture evidence. The appropriate status is OBSTRUCTED, not an extrapolated score.

7 The energy-quality frontier

7.1 Definition

Let $\mathcal{X}$ be the feasible configurations of a deployed system. A configuration includes model, quantization, decoding, batching, hardware, parallelism, prompt policy, tool scaffold, and serving load. For $x \in \mathcal{X}$, let $S_\mu(x)$ be expected task service, $E_\mu(x)$ expected energy, and $L(x)$ a latency statistic.

Definition 7.1 (Energy-quality frontier)**.** For energy budget e and latency limit $L^\star$, define

$$F_\mu(e, L^\star) = \sup_{x \in \mathcal{X}} \{S_\mu(x) : E_\mu(x) \le e,\ L(x) \le L^\star\}. \tag{4}$$

A configuration x dominates y when

$$E_\mu(x) \le E_\mu(y), \qquad S_\mu(x) \ge S_\mu(y),$$

with at least one strict inequality, and both satisfy the same latency constraint. Dominated points should not be recommended for that task distribution unless an omitted attribute explains the choice.

7.2 Why the frontier is more informative than one ratio

A ratio draws rays from the origin. It can prefer a low-energy, low-capability configuration even when a user requires higher absolute quality. It can also ignore a saturation region where additional energy produces little service. The frontier exposes both.

Three decision rules can be applied to the same frontier:

1. maximize service subject to an energy budget;
2. minimize energy subject to a service floor;
3. maximize a declared utility over service, energy, and latency.

Only the third collapses the frontier into a scalar, and it does so by adding a decision maker's preferences. The frontier itself remains a technical object.

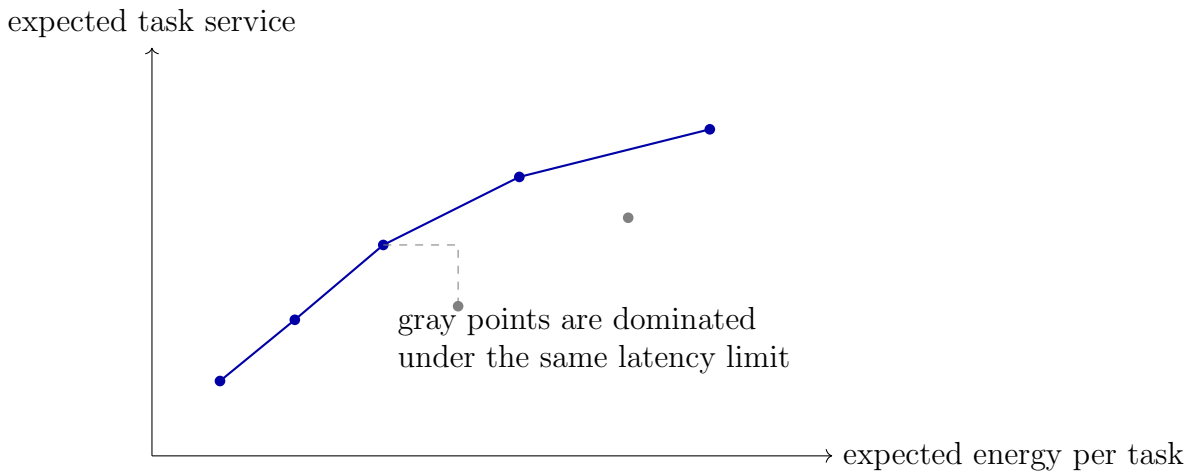

Figure 1: A schematic energy-quality frontier. The frontier preserves options for users with different budgets and quality requirements.

7.3 Frontier uncertainty

An empirical point is an estimate, not a fixed coordinate. Near-ties require joint uncertainty in service, energy, and latency. A conservative report can classify a point as robustly dominated only when its uncertainty region is dominated. Otherwise, display the overlapping regions and avoid a categorical claim.

Operating points also drift with load. Batch size can improve energy per task while increasing queueing latency. Hardware utilization can alter idle allocation. The frontier should therefore be indexed by arrival process and service policy, not just by a model name.

8 Uncertainty and inference

8.1 Sources of uncertainty

At least five uncertainties enter $\widehat{\text{QAIJ}}$:

1. task sampling from the target population;
2. stochastic model outputs and tool trajectories;
3. grading error or disagreement;
4. energy-meter calibration and sampling alignment;
5. task-distribution and shared-infrastructure weights.

Repeated runs on a fixed benchmark quantify output variation but not generalization to unseen tasks. More benchmark items quantify item variation but not meter bias. A single interval cannot hide which components were modeled.

8.2 Percentile bootstrap

The reference implementation uses the nonparametric bootstrap [10], resampling task attempts with replacement under a fixed seed. For bootstrap replicate b, it computes

$$\widehat{\theta}^{(b)} = \frac{\sum_i w_i^{(b)} s_i^{(b)}}{\sum_i w_i^{(b)} E_i^{(b)}}.$$

The interval takes empirical quantiles. This method is transparent and useful for fixtures, but its sampling unit must be correct. If attempts are nested within tasks and task families, the resampling should follow that hierarchy. METR uses a hierarchical bootstrap for its time-horizon analysis [6].

When denominator variation is large or near zero, ratio inference needs more care. Operational energy is positive and usually well separated from zero, but the raw values and denominator distribution should still be published.

8.3 Reliability intervals

For x binary successes in n trials, the Wilson interval is [11]

$$\frac{\widehat{p} + z^2/(2n) \pm z\sqrt{\widehat{p}(1-\widehat{p})/n + z^2/(4n^2)}}{1 + z^2/n}.$$

The code computes this interval using the standard library's normal distribution quantile. It avoids intervals below zero or above one.

For very small n, especially when $x = 0$ or $x = n$, report a continuity-corrected Wilson or exact-binomial interval as a sensitivity check. No interval method compensates for dependent attempts or an unrepresentative perturbation set.

Reliability is conditional on the tested perturbations. Repeating the same prompt at temperature zero tests less variation than changing tool state, network response, context order, and input paraphrase. The protocol should name the perturbation set.

8.4 Generalized performance

NIST AI 800-3 separates accuracy on a fixed benchmark from generalized accuracy over a superpopulation [5]. The same distinction applies to S_μ. A mixed-effects or item-response model can estimate latent system and item components, provided its assumptions are checked. The output should not be described as generalized if the target population is undefined.

For energy, generalization may require a separate load model. Energy per task at full batch and stable load may not transport to bursty production traffic. The target population therefore includes both tasks and system states.

8.5 Meter uncertainty

Let measured energy be

$$\widetilde{E}_i = E_i + \epsilon_i + \beta,$$

where ϵ_i is random measurement error and β is calibration bias. Repetition can reduce the first term but not the second. Calibration records and analyzer ranges matter.

Timestamp misalignment can attribute pre-run warm-up or post-run cooldown incorrectly. The power interval should be triggered by the benchmark harness and checked against request timestamps. If telemetry is used, sampling frequency and counter wraparound must be tested.

8.6 Multiple comparisons

Leaderboards compare many systems, scenarios, and operating points. Selecting the largest observed ratio creates winner's curse. Report all preregistered comparisons, intervals, and task strata. For procurement, a practically meaningful non-inferiority margin is often more useful than a claim that two close point estimates differ.

9 Measurement protocol

This section defines a minimum protocol for a publishable system-operational QAIJ result.

9.1 Step 1: declare the decision and estimand

State whether the study supports engineering optimization, model selection, capacity planning, or scientific comparison. Declare $\mathcal{A}$, the task population, the energy boundary, and the latency contract. Do not choose weights after observing rankings.

9.2 Step 2: freeze the system

Record model identifier and checksum when available, weights, tokenizer, quantization, numerical precision, inference engine, compiler, kernels, drivers, operating system, hardware, firmware, parallelism, sampling parameters, prompt template, tool scaffold, and network dependencies.

A provider model name is insufficient when endpoints can change silently. Record request dates, endpoint version fields, and behavioral fingerprints that licensing permits.

9.3 Step 3: define the task distribution

Publish task sources, inclusion rules, deduplication, contamination checks, family labels, sampling weights, and target population. Separate fixed benchmark inference from deployment generalization.

For interactive tasks, specify human actions, tool permissions, stopping rules, and whether human time is included in the service outcome. Human labor is not converted into joules in this metric.

9.4 Step 4: define difficulty

Choose strata before evaluation. If using expert completion time, publish the expert population, timing protocol, censoring, and uncertainty. If using item response theory, publish

model form, fit diagnostics, anchors, and scale identification. Always include unweighted results.

9.5 Step 5: define grading

Use executable tests where they faithfully express the task. For human or model judging, validate the rubric, blind system identity, randomize order, measure agreement, and adjudicate a sample. Archive raw outputs when privacy and licensing permit.

9.6 Step 6: measure reliability

Repeat task attempts across the variation relevant to deployment. A minimum count cannot be universal because task cost and expected failure rates differ. Publish the number of tasks, attempts per task, seeds, and transient failures.

9.7 Step 7: instrument energy

For system-operational claims, prefer synchronized wall power. Configure the analyzer range, log calibration, and record sampling rate. Device telemetry is reported as a diagnostic series, not silently substituted for wall energy.

For shared serving, define the allocation rule. A simple rule may assign dynamic energy to requests and allocate baseline idle energy by occupied time, but alternative rules should be tested. No rule is neutral when multiple tenants share hardware.

Report sensitivity to at least one credible alternative allocation rule when shared energy is material. Time-proportional and resource-proportional rules can yield different rankings under high concurrency. If task-level allocation cannot be supported from telemetry, report a pooled service-per-joule result for the shared system instead of invented precision.

9.8 Step 8: handle warm-up and idle energy

Declare whether compilation, model loading, cache warm-up, and idle periods are included. An engineering kernel benchmark may exclude setup. A deployed service study should include the setup and idle share that its operating model causes.

Report both gross and incremental energy when possible:

$$E_{\text{gross}} = \int_{\tau} P(t)\,dt, \qquad E_{\text{incremental}} = \int_{\tau} \max\{P(t) - P_{\text{idle}}, 0\}\,dt.$$

These answer different questions. The primary denominator must be named.

9.9 Step 9: preserve failures

Do not drop failed, refused, malformed, timed-out, or retried requests from the energy denominator. Record their failure class and energy. If the serving system retries automatically, the entire chain belongs to the attempt unless the functional unit says otherwise.

9.10 Step 10: control load and latency

Measure arrival process, concurrency, batch policy, queueing, time to first token where relevant, completion latency, and tail percentiles. Throughput comparisons under unmatched latency targets are not directly comparable.

9.11 Step 11: repeat and randomize

Randomize run order to reduce temperature, load, and drift confounding. Repeat across times or machines if the claim covers them. Monitor thermal throttling, background processes, and clock behavior. Preserve raw power traces and request logs.

9.12 Step 12: report numerator and denominator

Publish expected service, expected energy, their ratio, uncertainty, and every manifest field. A reader should be able to recompute the ratio and substitute an alternative task distribution without reconstructing hidden logs.

Table 2: Protocol audit checklist.

ID	Audit item	Pass condition
P1	Estimand	Decision, task population, latency target, and energy scope fixed
P2	System	Model, software, hardware, precision, and tokenizer recorded
P3	Tasks	Sources, weights, families, exclusions, and target population published
P4	Difficulty	Convention fixed, uncertainty reported, unweighted result retained
P5	Grading	Correctness and quality rules validated and reproducible
P6	Reliability	Repetitions and perturbation distribution match the claim
P7	Metering	Meter location, calibration, range, and synchronization recorded
P8	Shared load	Idle, warm-up, batching, and infrastructure allocation declared
P9	Failures	Failed and late attempts remain in denominator energy
P10	Service	Concurrency, latency, and retry policy fixed
P11	Inference	Sampling unit and uncertainty method match dependence structure
P12	Release	Raw aggregates, code, versions, and exclusions available

10 Reference implementation

10.1 Scope

The module `src/joule_standard/intelligence.py` uses only the Python standard library. It defines:

- `TaskResult`, with task, system, energy, latency, tokens, quality, correctness, reliability, usefulness, difficulty, and sampling weight;
- `quality_adjusted_service_per_joule`, implementing eq. (3);
- `bootstrap_qaij`, a seeded percentile bootstrap;
- `wilson_interval`, for binary reliability;
- `pareto_frontier`, for non-dominated energy-quality points;
- `rank_systems`, for explicit QAIJ or token-throughput comparisons.

The module validates probabilities, nonnegative weights, positive energy, and finite inputs. It does not estimate economic value, carbon emissions, or general intelligence.

10.2 Executable claims

The test file `tests/test_intelligence.py` maps paper claims to fixtures.

Table 3: Executable evidence and publication status.

Tested property	Status	Interpretation
Ratio of expected service to expected energy	COMPUTATIONAL	Implementation matches eq. (3)
Late attempts retain denominator energy	COMPUTATIONAL	Failure accounting follows protocol
Tokenizer changes token ratio only	COMPUTATIONAL	Fixture witnesses theorem 5.1
Token and quality rankings reverse	COMPUTATIONAL	Fixture implements strict reversal
Task-mix rankings reverse	COMPUTATIONAL	Fixture implements proposition 6.1
Seeded bootstrap is reproducible	COMPUTATIONAL	Same data and seed give same interval
Pareto filter removes dominated points	COMPUTATIONAL	Algorithm matches frontier definition
Wilson interval and input validation	COMPUTATIONAL	Reliability helper respects bounds

Passing tests do not prove external validity. They show that the reference calculation obeys the published definitions and counterexamples.

10.3 Ratio implementation

The estimator computes weighted service and energy separately:

$$\widehat{S} = \frac{\sum_i w_i s_i}{\sum_i w_i},$$
$$\widehat{E} = \frac{\sum_i w_i E_i}{\sum_i w_i},$$
$$\widehat{\mathrm{QAIJ}} = \frac{\widehat{S}}{\widehat{E}}.$$

For a late task, the service method returns zero before multiplying quality factors. The energy object remains unchanged.

10.4 Frontier implementation

The frontier routine first filters points that violate $L^\star$. It then marks candidate y dominated when another point x satisfies

$$E(x) \le E(y), \qquad S(x) \ge S(y)$$

with one strict inequality. Returned points are ordered by energy. This quadratic reference algorithm favors auditability over scale. Large leaderboards may use a sorted sweep after handling ties.

11 Worked examples

11.1 Example 1: same answer, different tokenizer

Suppose two tokenizers segment the same correct answer into 10 and 40 tokens. The measured system energy is 10 joules and every service factor equals one. Tokens per joule is 1 or 4, depending on tokenizer. QAIJ is 0.1 task equivalents per joule in both cases.

The example isolates tokenization. Real tokenizers can also affect model behavior because tokenization is part of the model. In that case, task outcome or energy may change, and the evaluator should measure those changes rather than assume invariance.

11.2 Example 2: late answers

Consider one successful answer using 5 joules in one second and one correct but late answer using 15 joules in three seconds. With $L^\star = 2$ seconds and equal task weights,

$$\widehat{S} = 0.5, \qquad \widehat{E} = 10 \text{ joules}, \qquad \widehat{\mathrm{QAIJ}} = 0.05.$$

Dropping the late answer would report 0.2, overstating deployed service by a factor of four. Counting its energy but not its service makes the contract visible.

11.3 Example 3: reliability

Suppose a tool-using agent succeeds in 8 of 10 independent runs. The point reliability is 0.8, but the uncertainty is wide. A Wilson interval at 95 percent is approximately 0.49 to 0.94. Reporting only 0.8 would create false precision.

If the ten attempts reuse the same hidden state or tool cache, they are not independent. The effective sample size is smaller. The interval method cannot repair a bad experimental design.

11.4 Example 4: difficulty-weight sensitivity

Let easy and hard tasks have unweighted service scores as in proposition 6.1. If a weighted scalar assigns the hard stratum five times the easy stratum, the hard specialist receives a larger composite gain. That result reflects the declared weight, not a discovery that a hard task has five times the intrinsic value.

Report the stratum vector first:

$$\left(\frac{S_e}{E_e}, \frac{S_h}{E_h}\right).$$

Then show any composite under at least two plausible normalizations. A reader can see whether the decision is robust or weight-driven.

11.5 Example 5: system and device boundaries

System A uses 70 joules at the accelerator and 30 joules elsewhere. System B uses 80 joules at the accelerator and 10 joules elsewhere. With equal service, device-only accounting ranks A first, while wall accounting ranks B first:

$$70 < 80, \qquad 100 > 90.$$

The result is another rank reversal, but its source is boundary choice rather than task mix. Neither number is wrong if labeled. Mixing the two in one table is wrong.

12 Interpretation and decision use

12.1 Engineering optimization

Within a fixed model, task distribution, and quality floor, QAIJ can compare quantization, caching, batching, kernel, and routing configurations. The frontier shows whether an optimization shifts the feasible set or merely moves along a latency tradeoff.

An optimization that raises tokens per joule but reduces correct service does not improve the quality-adjusted metric. An optimization that reduces energy while preserving task service and latency produces an unambiguous technical gain under the fixed boundary.

12.2 Model selection

For model selection, publish task-level results. The aggregate depends on μ, and users may have different mixes. A routing policy can dominate a single-model policy by sending easy cases to a small model and difficult cases to a larger one, but router errors and router energy belong in the system.

The selected configuration includes the entire policy:

$$x = (\text{router}, \text{models}, \text{serving stack}, \text{hardware}).$$

Reporting only the chosen model's energy omits the cost of selection.

12.3 Procurement

A procurement rule should use constraints before ratios. First set minimum correctness, reliability, safety, privacy, and latency. Then compare energy among qualifying systems or inspect the frontier. A high ratio from a system below the service floor is irrelevant.

Usefulness weights in procurement are stakeholder choices. Publish them and run sensitivity analysis. If stakeholder rankings conflict, keep the vector rather than invent a neutral average.

12.4 Capacity planning

Tokens remain useful for memory and throughput forecasts. QAIJ adds the missing service layer. A capacity model can retain both:

$$\frac{\text{tokens}}{\text{second}}, \quad \frac{\text{joules}}{\text{token}}, \quad \frac{\text{verified task service}}{\text{joule}}.$$

The three quantities answer production, energy, and outcome questions.

12.5 Scientific comparison

Scientific comparisons require a versioned harness, open task data where possible, raw outputs, power traces, and code. Closed endpoints can still be evaluated, but model drift and unavailable meter access limit reproducibility. The status of a cross-provider comparison may be CONDITIONAL or OBSTRUCTED.

13 Failure modes and adversarial behavior

13.1 Benchmark gaming

A system can optimize against the public task set without improving the target population. Contamination, memorization, evaluator exploitation, and grader-specific style can raise measured service. Holdout tasks, transcript audits, multiple graders, and post-deployment monitoring reduce this risk but do not eliminate it. The benchmark-lottery and general-benchmark critiques show why task selection and construct scope must remain visible [13, 14].

13.2 Verbosity gaming

Tokens-per-joule discourages verbosity only if energy grows faster than token count. Tokens-per-second may reward long easy continuations. A quality-adjusted task unit avoids direct reward for volume, but the quality grader may still prefer elaborate answers. Rubrics should reward required evidence and penalize irrelevant material.

13.3 Difficulty gaming

If difficulty is based on human time, tasks can be made cumbersome without becoming cognitively valuable. Bad tooling, unclear instructions, and setup latency can inflate the duration. Difficulty protocols should separate task substance from avoidable environment friction.

13.4 Reliability gaming

Retry-until-success policies can raise observed success if retries are hidden. The functional unit must state whether one user request includes all retries. Energy, latency, and failed attempts from the retry chain must remain in the record.

13.5 Boundary gaming

Device-only energy can make off-device preprocessing, retrieval, networking, or CPU work disappear. Facility multipliers can be selected from favorable periods. Lifecycle allocation can use optimistic utilization. A boundary manifest and bridge table are the defense.

13.6 Task-mix gaming

A provider can report a favorable mixture or exclude expensive failures. Versioned task weights and a public task-level dataset allow reweighting. When privacy prevents release, publish sufficient aggregate cells and an independent audit.

14 Limitations

Limitation 14.1 (No universal intelligence unit). The metric measures task service on a declared population. It does not identify general intelligence, consciousness, understanding, or a domain-independent capability scalar.

Limitation 14.2 (No realized economic value). Usefulness weights are evaluation inputs. They are not causal estimates of revenue, welfare, labor savings, scientific discovery, or avoided loss.

Limitation 14.3 (Difficulty is local). Human time and item-response scales depend on tasks, people, tools, and model assumptions. Cross-domain difficulty comparisons require an additional construct.

Limitation 14.4 (Operational energy is not lifecycle impact). System wall energy omits embodied hardware and may omit cooling, networking, and upstream energy. It also does not specify carbon, water, land, or local grid effects.

Limitation 14.5 (Aggregation embeds preferences). Any scalar combining quality, correctness, reliability, usefulness, and difficulty reflects a chosen aggregation rule. The vector and sensitivity results are part of the finding.

Limitation 14.6 (Benchmarks can fail to transport). Performance and energy can change under new tasks, load, hardware, software, and interaction patterns. Statistical precision on a fixed benchmark does not establish external validity.

Limitation 14.7 (Meter access is unequal). Closed providers may not expose wall energy. Modeled estimates should not be mixed with calibrated measurements without labels and uncertainty.

15 Research agenda

15.1 Shared evaluation datasets

The field needs task-level datasets that join raw outputs, executable or human grades, reliability repetitions, request timing, wall-power traces, hardware state, and task-family metadata. Privacy-preserving aggregate releases may be necessary for production workloads.

15.2 Hierarchical estimators

Future work should model tasks nested in families, attempts nested in tasks, and machines nested in sites. A joint model can separate task, model, and hardware variation. It should retain raw ratio components and pass posterior predictive checks.

15.3 Interactive systems

Human-AI workflows complicate the functional unit. The model may reduce human time but add review or correction. Energy-normalized technical service can be measured, but productivity and value require a randomized or credible counterfactual study of the combined workflow.

15.4 Dynamic routing

Routers can allocate tasks to models based on predicted difficulty. The evaluation must include router errors, abstentions, escalation, duplicate inference, cache behavior, and total system energy. A policy frontier may be more relevant than a model frontier.

15.5 Lifecycle extension

A lifecycle extension would allocate embodied accelerator, host, cooling, and building energy across service units. Utilization and lifetime assumptions will dominate some comparisons.

The operational metric should remain visible rather than being buried inside one lifecycle number.

15.6 Task service and causal value

The next paper in the AI sequence should estimate the link from verified task service to adopted actions and downstream effects. The bridge may use randomized deployments, encouragement designs, or careful observational methods. It should not convert QAIJ into dollars with a fixed coefficient.

16 Conclusion

Tokens per joule measures token production under a named tokenizer and serving configuration. It does not measure useful AI output. A defensible energy-normalized evaluation starts with a task population, service rule, latency contract, and physical boundary. It counts energy from failures and late attempts, reports numerator and denominator separately, and treats the result as distribution-specific.

Quality-adjusted task service per joule supplies that narrower object. Its algebra aggregates cleanly, its tokenization invariance is explicit, and its task-mix dependence is visible rather than hidden. The energy-quality frontier then preserves choices that a single ratio discards.

The framework does less than its title might suggest. That is deliberate. It does not identify general intelligence or realized economic value. It gives researchers and operators a reproducible way to ask a smaller question: under this task distribution, service rule, latency target, and energy boundary, how much verified AI task service did the measured system produce per joule?

A Formal assumptions and additional results

Assumption A.1 (Matched evaluation)**.** Systems compared directly use the same task distribution, grading rules, difficulty convention, usefulness weights, latency contract, energy boundary, and uncertainty design.

Assumption A.2 (Positive denominator)**.** Every evaluated system has finite positive expected energy under the target distribution.

Assumption A.3 (Measurable service)**.** Task service and energy are measurable under the evaluation's probability model.

Proposition A.4 (Currency and price independence)**.** *The technical* QAIJ *estimator is unchanged by currency rescaling because no monetary variable enters eqs.* (1) *and* (2)*.*

Proof. Immediate by inspection of the definitions. Difficulty, quality, correctness, reliability, usefulness, latency, and energy are the only arguments. Usefulness is dimensionless task-service weighting, not price. □

Proposition A.5 (Failure exclusion bias)**.** *Suppose at least one failed attempt has positive energy and zero service. Removing it from both numerator and denominator weakly increases the estimated* QAIJ*, with a strict increase whenever retained service is positive.*

Proof. Let retained totals be $S > 0$ and $E > 0$, and failed energy be $F > 0$. The complete estimate is $S/(E + F)$, while the filtered estimate is S/E. Since $E < E + F$, $S/E > S/(E + F)$. □

Proposition A.6 (Dominated-point irrelevance)**.** *Under a fixed task distribution and latency limit, if configuration x uses no more energy and supplies at least as much service as y, with one strict inequality, then y cannot solve either the minimum-energy problem at a service floor met by x or the maximum-service problem at an energy budget that admits y.*

Proof. For minimum energy, x meets every service floor met by y and has no greater energy. For maximum service, every budget admitting y also admits x, which has no lower service. The strict inequality prevents y from being uniquely optimal. □

Remark A.7 (Safety constraints). Safety, privacy, and security should usually be feasibility constraints or separate outcome axes. Folding severe harms into a smooth average can allow high ordinary-task performance to compensate for an unacceptable failure.

B Reproducibility schema

The following schema is a conceptual minimum. A concrete release may encode it as JSON, CSV plus metadata, or a relational dataset.

Table 4: Recommended reproducibility fields.

Field group	Contents
Run identity	Run UUID, timestamp, site, operator, code revision
Model	Provider, model, version, weights hash, tokenizer, license
Serving	Engine, precision, quantization, batch policy, concurrency
Hardware	Accelerator, host, memory, interconnect, firmware, clocks
Software	OS, driver, compiler, kernels, container, dependency lock
Task	Task ID, family, source, split, target weight, difficulty stratum
Prompt	System prompt, user prompt, tool policy, context, randomization
Output	Raw output or digest, token counts by named tokenizer, termination
Grade	Quality, correctness, grader, rubric, judge prompt, adjudication
Reliability	Attempt ID, seed, perturbation class, success definition
Latency	Queue, first response, completion, timeout, deadline indicator

Field group	Contents
Energy	Meter, location, sampling rate, calibration, start and stop times
Boundary	Device, system, facility, lifecycle, idle and warm-up allocation
Failure	Crash, refusal, timeout, invalid format, tool failure, retry chain
Uncertainty	Sampling unit, weights, interval method, confidence level
Exclusions	Unmetered components, unavailable outputs, privacy redactions

C Audit questions for reviewers

1. Does the title's word "intelligence" remain operationally qualified as task service throughout the paper?
2. Can every result be reconstructed from a declared task distribution?
3. Are difficulty and usefulness treated as declared constructs rather than natural quantities?
4. Do correctness and reliability avoid double counting?
5. Does every failed or late attempt retain denominator energy?
6. Are wall, device, facility, and lifecycle energy kept separate?
7. Is a tokenizer named wherever token throughput is reported?
8. Do confidence intervals use the independent sampling unit?
9. Are benchmark and generalized performance distinguished?
10. Are frontier comparisons made under a matched latency constraint?
11. Does any sentence imply realized economic value without a causal design?
12. Are code tests described as computational evidence rather than proof?

D Nonclaims

For avoidance of doubt, this paper does not claim:

- that a benchmark score is general intelligence;
- that difficult tasks are intrinsically more valuable;
- that human completion time is comparable across every domain;
- that correctness, quality, reliability, and usefulness are naturally independent;
- that a model's service score transports to a new deployment;
- that operational electricity is lifecycle energy or environmental impact;
- that efficient inference reduces total data-center energy demand;
- that technical task service is revenue, welfare, or productivity;
- that one scalar should replace task-level and frontier reports;
- that tests of the reference implementation formally verify deployed AI systems.

References

[1] MLCommons. MLPerf Inference documentation. https://docs.mlcommons.org/inference/, accessed July 2026.

[2] MLCommons. Power measurement for MLPerf Inference. https://docs.mlcommons.org/inference/power/, accessed July 2026.

[3] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and others. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. https://openreview.net/forum?id=iO4LZibEqW.

[4] National Institute of Standards and Technology. AI test, evaluation, validation, and verification. https://www.nist.gov/ai-test-evaluation-validation-and-verification-tevv, accessed July 2026.

[5] Drew Keller, Kweku Kwegyir-Aggrey, Ryan Steed, Anita K. Rao, Julia L. Sharp, and A. Stevie Bergman. Expanding the AI evaluation toolbox with statistical models. NIST AI 800-3, 2026. https://doi.org/10.6028/NIST.AI.800-3.

[6] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, and others. Measuring AI ability to complete long tasks. arXiv:2503.14499, 2025. https://arxiv.org/abs/2503.14499.

[7] METR. Task-completion time horizons of frontier AI models. https://metr.org/time-horizons/, accessed July 2026.

[8] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54 to 63, 2020. https://doi.org/10.1145/3381831.

[9] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1 to 43, 2020. https://jmlr.org/papers/v21/20-312.html.

[10] Bradley Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1):1 to 26, 1979. https://doi.org/10.1214/aos/1176344552.

[11] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209 to 212, 1927. https://doi.org/10.1080/01621459.1927.10502953.

[12] Georg Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests.* Danish Institute for Educational Research, Copenhagen, 1960.

[13] Mostafa Dehghani, Yi Tay, Alexey Gritsenko, Zhe Zhao, Neil Houlsby, Fernando Diaz, Donald Metzler, and Oriol Vinyals. The benchmark lottery. arXiv:2107.07002, 2021. https://arxiv.org/abs/2107.07002.

[14] Inioluwa Deborah Raji, Emily Denton, Emily M. Bender, Alex Hanna, and Amandalynne Paullada. AI and the everything in the whole wide world benchmark. In *NeurIPS Datasets and Benchmarks*, 2021. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/084b6fbb10729ed4da8c3d3f5a3ae7c9-Abstract-round2.html.