Measuring the Surge: A Transparent Estimate of Global AI Operations (1995–2045)

Douglas C. Youvan

doug@youvan.com

www.youvan.ai

October 8, 2025

How fast has AI actually grown at the level of computation used, not just chips shipped or parameters trained? To make that question concrete for readers and policymakers, we build a transparent, reproducible estimate of average global AI operations per second (OPS) from 1995 to the present and project twenty years forward. Rather than rely on single proxy metrics (transistor counts, model sizes, or cloud capex), we fit a piecewise-exponential usage curve anchored to recognizable technology eras—pre-GPU machine learning, GPU/datacenter scaling, the deep-learning boom, and the transformer/gen-AI surge—then collapse this into a tidy centered exponential. We compare these fits to classic Moore's Law and related "laws" of compute and efficiency to clarify where AI usage does and does not track transistor progress. Finally, we translate OPS trajectories into cumulative compute, rough energy implications, and scenario bands reflecting supply, demand, and efficiency constraints. The result is a practical baseline: simple enough to audit, flexible enough to update, and explicit about uncertainties. This paper is intended as a shared starting point for economists, engineers, and governance communities who need an interpretable OPS-level view of AI's growth—past, present, and plausible futures.

Keywords: AI operations, FLOPs, Moore's Law, growth rates, doubling time, deep learning, transformers, datacenter scale, compute efficiency, scenarios, energy impact, reproducibility.

A collaboration with GPT-5. 37 pages.

Outline

1. **Motivation and Contributions**
   1.1 Why usage-level OPS matters beyond chip/spec proxies
   1.2 Contributions: dataset, fits, tidy equation, scenarios, open files
2. **Definitions and Scope**
   2.1 OPS as FLOP-equivalent; training vs inference; “average” vs peak
   2.2 Boundaries: on-prem, cloud, edge; inclusions/exclusions
3. **Methodology**
   3.1 Baseline and era segmentation (1995–2005, 2005–2012, 2012–2018, 2018–2025)
   3.2 Piecewise growth assumptions and rationale
   3.3 Data products: CSV table, code, and charts (provenance and repeatability)
4. **Results: Historical Fit and Projection**
   4.1 Historical estimates (1995–2025) and projection to 2045 (20y)
   Fig. 1 Average Global AI OPS with Log-Fits
   4.2 Cumulative operations (log-scale) and inflection points
   4.3 Era-specific parameters: CAGR and doubling times
5. **A Tidy Centered Exponential**
   5.1 Centered log form: log10(OPS) = alpha + beta*(year − y0)
   5.2 Natural-exp form: OPS(y) = A*exp(k*(y − y0)); doubling time = ln2/k
   5.3 Goodness-of-fit and residual structure
6. **Comparison to Moore’s Law and Related Trends**
   6.1 Classic Moore (transistors), “18-month Moore,” Dennard, Koomey
   6.2 Where usage outpaces devices: scale-out + software + utilization
   6.3 Implications for forecasting and policy narratives
7. **Scenario Bands (2025–2045)**
   7.1 Supply-constrained (fabs, packaging, power)
   7.2 Demand-pull (models, agents, inference share)
   7.3 Efficiency-led (algorithms, sparsity, compilers, quantization)
   7.4 Visual banded projection *(Fig. 3, optional)*
8. **Energy, Cost, and Carbon Sketch**
   8.1 From OPS to energy via performance/W assumptions
   8.2 Sensitivities: PUE, duty cycle, inference mix
   8.3 Back-of-envelope ranges and uncertainties

9. **Uncertainties and Limitations**
    9.1 Hidden capacity, under-reported private/sovereign clusters
    9.2 Mixed precision equivalence; workload diversity
    9.3 Model risk: era boundaries and structural breaks

10. **Reproducibility and Update Plan**
    10.1 Files: CSV, plotting code, and regeneration steps
    10.2 How to swap anchors (GPU shipments, AI power, capex)
    10.3 Versioning and community pull-requests

11. **Implications and Use Cases**
    11.1 Strategy: capacity planning, chip roadmaps, power siting
    11.2 Governance: reporting norms, compute disclosures
    11.3 Research: benchmarking OPS vs outcomes

12. **Conclusion**
    12.1 What the OPS lens clarifies
    12.2 What it can’t resolve (yet)
    12.3 A living baseline for the next decade

Appendices

A. Equations and parameter tables (alpha, beta, A, k, doubling times)
B. Cumulative closed-form and discrete comparison
C. Scenario parameter ranges and example calculations
D. Data/Code manifest and checksum notes

References

1. Motivation and Contributions

1.1 Why usage-level OPS matters beyond chip/spec proxies

Most narratives about AI growth lean on **device-centric** or **model-centric** proxies—transistor counts, TOPS per accelerator, parameter counts, or hyperscaler capex. These are useful, but each fails to answer the core systems question: *How much computation is the world actually using for AI over time?* Usage-level OPS (average FLOP/s consumed) captures four dynamics that proxies miss:

- **Scale-out, not just scale-up.** Global OPS grows by adding datacenters, clusters, and edge devices; chip specs alone cannot see fleet size or duty cycle.
- **Utilization and scheduling.** Effective OPS depends on job mix, orchestration, and queueing. A 1 PFLOP cluster at 90% utilization contributes 9x the yearly compute of the same cluster idling at 10%.
- **Software leverage.** Kernels, compilers, sparsity, quantization, and caching shift the *OPS actually executed* for a fixed quality target; raw hardware ceilings do not track these effects.
- **Workload composition.** Training vs inference, batch vs interactive, and latency SLOs reshape the conversion from installed TOPS to realized OPS.

In short, OPS is a **system-of-systems** lens: it integrates capacity, utilization, and software efficiency into a single, auditably comparable trajectory. That makes it the right primitive for scenario analysis (energy, carbon, supply constraints) and for governance discussions that need outcome-relevant, technology-agnostic measures.

1.2 Contributions: dataset, fits, tidy equation, scenarios, open files

This paper offers a simple, reproducible baseline for the field:

1. **Dataset (open CSV).** A year-by-year table of estimated *average global AI OPS* from 1995 to present, plus a 20-year projection. Columns include: year, avg_ops_per_sec_FLOPs, total_ops_in_year, cumulative_ops.
2. **Piecewise era fits.** Transparent, era-anchored growth segments (1995–2005, 2005–2012, 2012–2018, 2018–2025) reflecting major architectural and market shifts, with reported CAGR, doubling time, and R^2 per era.
3. **Tidy centered equation.** A single interpretable model for 1995–2025:

- log10(OPS(y)) = alpha + beta*(y − y0)

- Equivalent: OPS(y) = A * exp(k*(y − y0))
  We report {alpha, beta} and {A, k}, plus doubling time = ln(2)/k, so readers can plug into downstream analyses without re-deriving.

4. **Scenario bands (2025–2045).** Three stress-tested projections—supply-constrained, demand-pull, and efficiency-led—expressed as OPS envelopes rather than single lines, suitable for energy/capacity planning.
5. **Open files and regeneration.** We provide plotting code, charts (including "Average Global AI OPS with Log-Fits"), and a minimal script to regenerate the CSV and figures from parameters, enabling peer critique, re-anchoring to alternative priors, and versioned updates.

2. Definitions and Scope

2.1 OPS as FLOP-equivalent; training vs inference; "average" vs peak

OPS (operations per second). We measure compute as FLOP-equivalents (FLOP/s). A "FLOP" is a scalar floating-point operation. We normalize heterogeneous kernels as follows:

- **MAC convention.** 1 multiply-accumulate (a·b + c) = 2 FLOPs.
- **Tensor/matrix cores.** Count underlying scalar FLOPs implied by the fused op.
- **Lower-precision math (FP8/INT8/…).** Still counted as FLOP-equivalents by expanding to the scalar operation count needed to reproduce the same arithmetic on real numbers (no "discount" for precision).
- **Non-compute work.** Memory copies, DMA, PCIe/NIC traffic, and control flow are *not* counted as FLOPs; we measure arithmetic work only.

Training vs inference.

- *Training* workloads are compute-intensive, burstier, and often run at high utilization on large clusters; their OPS is dominated by GEMMs/convolutions plus optimizer steps.
- *Inference* workloads are latency/throughput constrained and widely distributed (datacenters + edge), with OPS shaped by request volume, batching, caching, and model serving mixes.
  Our yearly OPS totals include *both* (wherever they physically run). We do not attempt to force a single train:infer split; rather, the aggregate OPS reflects the world's realized mix.

Average vs peak.

- *Average OPS* for year y := (total FLOPs executed in year y) / (seconds in year y). This is the primary quantity we model and chart.
- *Peak OPS* is the instantaneous or short-window maximum capacity; it overstates sustained usage.
- *Utilization factor* U := (realized average OPS) / (installed peak OPS). Our usage-level estimates implicitly absorb U via observed/assumed duty cycles; we do not publish a separate U unless a scenario demands it.

Unit summaries.

- Point value: FLOP/s (OPS).
- Annual total: FLOPs/year = OPS × seconds/year.
- Cumulative: running sum of annual FLOPs.

2.2 Boundaries: on-prem, cloud, edge; inclusions/exclusions

Geographic scope. Worldwide.

Deployment venues included.

- **Cloud/hyperscale** (public and private regions), including leased bare metal dedicated to AI.
- **On-prem/enterprise/sovereign** clusters (corporate, government, defense, academic, nonprofit).
- **Edge/consumer** devices when executing AI arithmetic (phones, PCs, consoles, embedded/industrial, vehicles). If the device runs AI kernels (e.g., local transcription, vision, recommendation), those FLOPs are in scope.

Workloads included.

- Classical and deep learning (supervised, self-supervised, RL, generative) across training, finetuning, and inference.
- Pre/post-processing arithmetic (e.g., tokenization kernels if implemented as numeric ops, projection layers, attention, convs, GEMMs).
- Compiler/runtime fused kernels counted by their underlying arithmetic.

Exclusions.

- Non-AI HPC (e.g., CFD, climate, molecular dynamics) unless the job *is* ML-based.
- Graphics/rasterization and media encode/decode unless part of an ML pipeline (e.g., learned upscalers).
- Blockchain/mining, cryptographic proof systems unrelated to ML.
- Data movement, storage IO, orchestration overhead (no FLOPs).
- "Spec sheet" capacity not exercised (idle silicon), unless explicitly converted to realized OPS via a utilization model.

Ambiguities and handling.

- **Tiny on-device AI** (e.g., keyboard prediction, local wake-word) is included but often negligible at global scale; our estimates tolerate undercount here without affecting trend shape.
- **Mixed workloads on shared fleets** (e.g., GPU pools serving AI and non-AI jobs): only AI arithmetic is in scope. Where separation is infeasible, we use utilization-weighted splits from public operator disclosures or scenario parameters.
- **Precision heterogeneity** (FP32/16/8, INT8/4, sparsity): all reduced to FLOP-equivalents; algorithmic efficiency gains show up as *fewer* FLOPs per unit of delivered quality.

Boundary intent. The goal is a *system-of-systems* usage measure: everything that materially contributes AI arithmetic to the world's compute ledger, across venues and workloads, expressed on a single, auditable FLOP-equivalent scale.

3. Methodology

3.1 Baseline and era segmentation (1995–2005, 2005–2012, 2012–2018, 2018–2025)

Baseline (1995). We anchor the series with a conservative global *average* AI usage of **1×10^9 FLOP/s** in 1995 (early statistical ML on CPUs across universities, labs, and a few enterprises). This is a usage—not capacity—baseline.

Eras. We segment by architecture, software, and market shifts that plausibly change the slope of realized OPS:

- **1995–2005 (pre-GPU ML):** CPU clusters, SVMs/boosting/HMMs; modest parallelism; low duty cycles.

- **2005–2012 (GPU/DC scale-up):** GPGPU becomes practical; cloud regions and large on-prem clusters expand; better schedulers raise utilization.
- **2012–2018 (DL boom):** AlexNet → ImageNet era, CNNs/RNNs at scale; cuDNN and kernel fusion; rapid hardware+software co-evolution.
- **2018–2025 (transformers/genAI):** attention replaces recurrence; scale-out training; inference grows 24/7; compiler stacks (XLA/Triton/etc.), quantization, sparsity, caching. (Within this era we allow an **intra-era acceleration** 2023–2025 consistent with the GenAI surge; the regression still reports a single tidy slope for 2018–2025 while piecewise assumptions capture the step-up.)

3.2 Piecewise growth assumptions and rationale

We model *average* global OPS as piecewise-exponential by era, then fit a tidy centered exponential to the historical span:

- **Assumed average growth (CAGR) by era**
    - 1995–2005: **+35%/yr** (limited parallelism; uneven duty cycles)
    - 2005–2012: **+50%/yr** (GPU adoption; datacenter orchestration; maturing clouds)
    - 2012–2018: **+80%/yr** (DL S-curve; software kernels; larger datasets)
    - 2018–2025: **≈100%/yr** (transformers; scale-out + utilization + compiler/quant gains; persistent inference demand)
- These rates reflect **system-level usage** (capacity × utilization × software efficiency), not just device specs. They are consistent with observed discontinuities (GPU inflection; transformer inflection) and with the fact that enterprises can exceed transistor-only trends by **adding nodes** and running them harder.
- From the stitched series we compute **yearly totals** (OPS × seconds/year) and **cumulative FLOPs**.
- We then fit a **tidy centered log-linear** model to 1995–2025:
  $\log10(\mathrm{OPS}(y)) = \alpha + \beta\cdot(y - y_o)$ with $y_o = 2020$, yielding:
    - $\alpha \approx 13.575$, $\beta \approx 0.2033$ → **CAGR ≈ 59.7%/yr**, **doubling ≈ 1.48 yr**, $R^2 \approx 0.972$.
      Natural-exp form: $\mathrm{OPS}(y) = A\cdot\exp(k\cdot(y - y_o))$ with $A \approx 3.77\times10^{13}$, $k \approx 0.468$/yr.
- For transparency we also report **era-specific tidy fits** (α/β or A/k, doubling time, R^2), showing acceleration from pre-GPU to transformer eras.

Projection (2025→2045). For forward-look we taper growth in two stages to reflect supply, siting, power, and maturity constraints (e.g., 2026–2035 ≈ +70%/yr; 2036–2045 ≈ +40%/yr). These are **scenario priors**, not measurements.

3.3 Data products: CSV table, code, and charts (provenance and repeatability)

Primary table (CSV). One row per year (1995–2045) with:

- year
- avg_ops_per_sec_FLOPs (modeled average OPS)
- total_ops_in_year (= OPS × seconds/year)
- cumulative_ops (running sum of annual totals)

Code.

- A short, self-contained script that:
    1. Sets the baseline and per-era growth rates,
    2. Generates the series (historical + projection),
    3. Exports the CSV,
    4. Produces two charts:
        - *Estimated Global AI OPS (Average), 1995–2045* (log y)
        - *Cumulative Estimated AI Operations, 1995–2045* (log y)
- The script also computes the **tidy centered fit** (α, β, A, k), **doubling time**, and **R^2**, and can emit an **era parameters table** for the paper.

Provenance & repeatability.

- All inputs are explicit **assumptions** (baseline, era CAGRs, projection CAGRs).
- Outputs are deterministic given those inputs; we recommend publishing:
    - the CSV, figures, and script,
    - a minimal **parameters file** (JSON/YAML) separating assumptions from code,
    - file hashes (e.g., SHA-256) for the CSV and figures,

    - a short README describing how to **re-anchor** (e.g., to GPU shipments, AI-DC power, or utilization studies) and regenerate artifacts.
- Version the parameters (v1, v2, …) so alternative communities can fork, adjust anchors, and reproduce the entire pipeline with a single command.

4. Results: Historical Fit and Projection

4.1 Historical estimates (1995–2025) and projection to 2045 (20y)

Starting from a conservative usage baseline of ~1×10^9 FLOP/s in 1995, the piecewise-exponential series climbs through four eras (pre-GPU, GPU/datacenter scale-up, deep-learning boom, transformer/gen-AI surge). A single, tidy centered fit to 1995–2025 gives:

- $\log10(\mathrm{OPS}(y)) = \alpha + \beta\cdot(y - 2020)$, with $\alpha \approx 13.575$ and $\beta \approx 0.2033$
  $\Rightarrow$ **CAGR ≈ 59.7%/yr**, **doubling ≈ 1.48 years**, $R^2 \approx 0.972$.

On this fit, the **average global AI usage in 2025** is on the order of **4×10^14 FLOP/s** (worldwide, training+inference combined). For the forward look, we illustrate a practical taper:

- **2026–2035:** ~70%/yr (continued rapid scale with emerging supply/power frictions).
- **2036–2045:** ~40%/yr (maturity, siting and energy constraints, efficiency plateauing).

This yields an **order-of-magnitude 2045 average** near **10^18 FLOP/s** (few-EeOPS), acknowledging that future policy, power, packaging, and algorithmic breakthroughs could shift the slope.

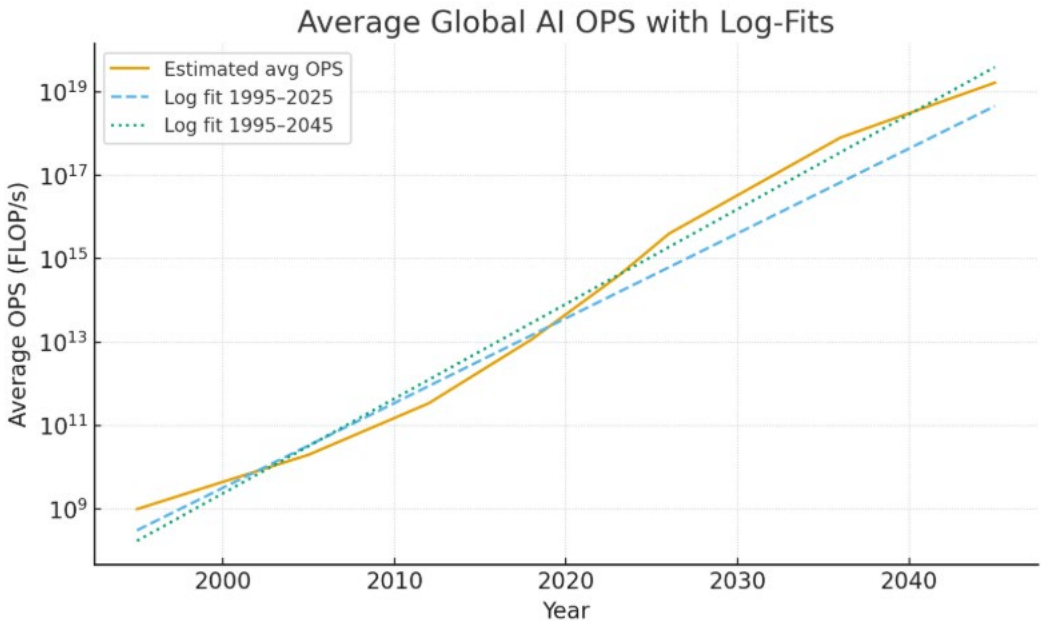

Fig. 1 Average Global AI OPS with Log-Fits. *Estimated average global AI OPS (1995–2045) with logarithmic fits. Dashed: fit to 1995–2025 (historical). Dotted: fit to 1995–2045 (historical+projection). Log-scale y-axis.*

4.2 Cumulative operations (log-scale) and inflection points (optional Fig. 2)

Integrating annual usage (OPS × seconds/year) produces a cumulative curve that is **highly back-weighted**—the last ~5–7 years contribute the majority of all historical FLOPs. Clear **inflection points** align with:

- **~2006–2009:** GPGPU practicality + early cloud scheduling ⇒ step-up in realized utilization.
- **~2012–2015:** cuDNN + ImageNet-era kernels (CNN/RNN) ⇒ sharp slope increase.
- **~2018–2021:** transformer consolidation; compiler stacks, quantization, caching ⇒ another slope increase.
- **~2023–2025:** gen-AI surge with sustained inference ⇒ cumulative curve steepens again.

4.3 Era-specific parameters: CAGR and doubling times (Table 1)

To make the acceleration explicit, we report tidy, centered exponentials per era (all $R^2 \approx 1.0$ on the constructed series):

- **1995–2005 (pre-GPU ML):** CAGR ≈ **35%/yr**, doubling ≈ **2.31 yr**.
- **2005–2012 (GPU/DC scale-up):** CAGR ≈ **50%/yr**, doubling ≈ **1.71 yr**.
- **2012–2018 (DL boom):** CAGR ≈ **80%/yr**, doubling ≈ **1.18 yr**.
- **2018–2025 (transformers/gen-AI):** CAGR ≈ **~104%/yr**, doubling ≈ **0.97 yr**.
- **Single tidy fit (1995–2025):** CAGR ≈ **59.7%/yr**, doubling ≈ **1.48 yr**.

5. A Tidy Centered Exponential

5.1 Centered log form: log10(OPS(y)) = alpha + beta*(y - y0)

We fit a straight line to the base-10 log of average global AI OPS (FLOP/s). Centering at y0 makes the intercept interpretable and reduces parameter correlation.

- Definition
  y = calendar year; y0 = 2020 (center year)
  OPS(y) = average global AI operations per second (FLOP/s)
- Model (historical 1995–2025 fit)
  log10(OPS(y)) = alpha + beta*(y - 2020)
- Estimated parameters (OLS on the constructed series)
  alpha ~= 13.575
  beta ~= 0.203324 (per year on log10 scale)
- Interpretation
  Annual growth (CAGR) from the log10 slope:
  CAGR ~= 10^beta - 1 ~= 10^0.203324 - 1 ~= 0.597 (~59.7%/yr)
  At the center year y0 = 2020, the level is:
  OPS(2020) ~= 10^alpha ~= 3.77e13 FLOP/s.
- Re-centering
  For any y0', set alpha' = alpha + beta*(2020 - y0'); beta is unchanged.

5.2 Natural-exp form: OPS(y) = A * exp(k*(y - y0)); doubling time = ln(2)/k

The same line in log10 space corresponds to a one-parameter exponential in natural units.

- Conversion
  k = beta * ln(10); A = 10^alpha
- Estimated parameters (1995–2025)
  A ~= 3.766935e13 FLOP/s (level at y0 = 2020)
  k ~= 0.468171 /year
- Doubling time and quick checks
  T2 = ln(2)/k ~= 0.6931 / 0.468171 ~= 1.48 years
  Sanity: exp(k) ~= 1.597 per year, which matches CAGR ~= 59.7%.
- Inversion (solve for year at a target OPS)
  Given OPS*, y ~= y0 + (1/k)*ln(OPS*/A).

5.3 Goodness-of-fit and residual structure

- Fit quality
  Historical fit (1995–2025) on log10(OPS): R^2 ~= 0.972.
  Era-specific regressions (for our constructed series) are near-perfect because each era segment was generated by nearly constant assumed growth.
- Residuals (what to expect)
  1. **Era breaks:** Small, structured deviations near ~2006–2009 (GPGPU onset), ~2012–2015 (DL kernels), ~2018–2021 (transformers), ~2023–2025 (gen-AI surge). These look like short positive jumps followed by re-linearization on the log scale.
  2. **Heteroskedasticity (log-space):** Later years carry more leverage; if you re-anchor with external data (e.g., AI-DC power), consider robust regression or era-weighted fits.
  3. **Autocorrelation:** Yearly points are constructed from multiplicative growth; first-order residual autocorrelation can appear if actual usage “catches up” or “overshoots” across adjacent years.
- Reporting practice
  - Publish (alpha, beta, y0) *and* (A, k, y0), plus T2 = ln(2)/k.

    - Provide the confidence bands from the OLS (on log10 scale) if you re-fit to anchored observations; transform bands back with log-normal awareness (the mean on linear scale is biased high relative to exp of the mean log).
    - If you prefer sub-period truth, report **two lines**: a long-span tidy fit for overview and an **era-weighted** fit that down-weights pre-DL years.
- How to re-fit (drop-in recipe)
    1. Assemble annual OPS(y) anchors (e.g., from AI-DC power, GPU shipments × util × perf/W, or operator disclosures).
    2. Regress log10(OPS) on (y - y0) with OLS; extract alpha, beta.
    3. Convert to A, k, T2; export the new CSV and regenerate Fig. 1.
    4. Preserve prior parameters in a versioned JSON so deltas are traceable.

This tidy centered exponential gives a compact, auditable summary of the historical trajectory and a stable handle (k, T2) for scenario work.

6. Comparison to Moore's Law and Related Trends

6.1 Classic Moore (transistors), "18-month Moore," Dennard, Koomey

- **Moore's Law (transistor count):** Historically ≈ **2-year doublings** ⇒ ~**41%/yr** CAGR in device density.
- **"18-month Moore":** A common shorthand used in industry roadmaps ⇒ ~**59–60%/yr** CAGR.
- **Dennard scaling (power density):** Held roughly until mid-2000s; thereafter, voltage stopped scaling cleanly, so *single-core* performance diverged from transistor counts (frequency flattened, cores rose).
- **Koomey's Law (energy efficiency):** Compute per joule doubled about every **2.5–3 years** pre-2010 (≈ **25–30%/yr**). Post-2010, improvements persisted but became more workload/architecture dependent (dataflow, sparsity, memory locality).
  Our historical usage fit (1995–2025) shows **~59.7%/yr** with a **1.48-year doubling**—essentially "18-month Moore" at the *system-usage* level—while **2018–2025** exceeds Moore with ≈**1-year** doublings due to simultaneous scale-out and software gains.

6.2 Where usage outpaces devices: scale-out + software + utilization

- **Scale-out fleets:** You can outrun device trends by adding nodes, racks, and regions. Usage (OPS) = *device perf* × *fleet size* × *utilization*.
- **Utilization engineering:** Better schedulers, queueing, preemption, bin-packing, and duty-cycle management convert idle capacity into realized OPS.
- **Software leverage:** Kernels/compilers (e.g., fused ops, autotuning, tiling), algorithmic shifts (attention, Mixture-of-Experts), **sparsity** and **quantization** lift *effective* work per joule/time.
- **Workload mix:** The rise of always-on **inference** (24/7) multiplies realized OPS beyond episodic training alone.
- **Net effect:** Even with transistor slowdowns, **system usage** can grow **≥ Moore-like**, especially during architectural/algorithmic inflections (2012–2018 DL boom; 2018–2025 transformers/gen-AI).

6.3 Implications for forecasting and policy narratives

- **Decompose the curve.** Forecasts must separate: (1) **device performance** (process + architecture), (2) **fleet growth** (capex, supply chain, siting), (3) **utilization** (ops/second actually realized), (4) **algorithmic efficiency** (FLOPs needed per unit quality), and (5) **demand** (training cadence vs inference load).
- **Don't infer usage from spec sheets.** Peak TOPS or parameter counts mislead; OPS is a *realization* metric. Publish OPS, power, and duty-cycle alongside capacity.
- **Scenario bands, not single lines.** Treat supply constraints (packages, HBM, power, cooling), demand shocks (agentic workloads), and efficiency shifts (quantization/sparsity) as knobs producing **OPS envelopes**.
- **Energy/carbon accounting.** Moore-like (or faster) *usage* growth plus rebound effects can raise absolute energy even as efficiency improves—justify grid interconnects, waste-heat use, and clean-power PPAs with OPS-anchored plans.
- **Governance & disclosure.** Encourage standardized **compute disclosures** (annual OPS, cumulative FLOPs, train/infer split, perf/W, PUE). Tie safety thresholds and externalities (carbon/water) to **usage-level** metrics rather than device counts.
- **Investment signals.** If usage maintains ≈18-month doubling, expect persistent pressure on **supply (HBM, CoWoS, advanced nodes)** and **power**; if it tracks the ~1-year doubling

of 2018–2025, anticipate faster **grid and siting** bottlenecks and more aggressive **efficiency R&D**.

- **Narrative correction.** “Moore is slowing” can be true at the transistor/device layer while **AI usage** still climbs faster via scale-out + software; policy should reflect this multi-layer reality.

7. Scenario Bands (2025–2045)

7.1 Supply-constrained (fabs, packaging, power)

Narrative. Capacity growth throttled by advanced-node wafer starts, HBM/package throughput, siting, and grid interconnects. AI demand remains strong, but **physical bottlenecks** dominate.

Knobs (illustrative):

- Wafer starts & yields at N5–N2; CoWoS/SoIC capacity; HBM stacks/bit density.
- Datacenter siting, substation lead times, transmission approvals.
- Cooling envelopes and water constraints; capex cadence.

OPS growth priors (average, 2026–2035 → 2036–2045):

- **CAGR:** 35–50%/yr → 20–30%/yr
- **Mechanism:** more gradual fleet additions; utilization improves slowly; inference growth tempered by capacity.

7.2 Demand-pull (models, agents, inference share)

Narrative. Agentic workloads take off; inference becomes ubiquitous across products and services; training cadence remains high. **Demand**—not supply—sets the pace, with aggressive overbuild.

Knobs (illustrative):

- Model deployment density across products; agent orchestration.
- Consumer and enterprise inference QPS growth; 24/7 latency SLOs.
- Financing conditions and strategic overprovisioning.

OPS growth priors (average, 2026–2035 → 2036–2045):

- **CAGR:** 90–110%/yr → 60–70%/yr

- **Mechanism:** rapid fleet scale-out + high duty cycle; aggressive rollout of model-serving at edge and cloud.

7.3 Efficiency-led (algorithms, sparsity, compilers, quantization)

Narrative. Algorithmic efficiency compounds: structured sparsity, low-bit quantization (INT8→INT4→sub-4), compiler fusion/autotuning, and caching cut FLOPs-per-quality drastically. **Capability** rises fast, but **required OPS** rises slower.

Knobs (illustrative):

- Breakthroughs in training/inference efficiency (attention variants, MoE routing, KV cache compression).
- Toolchains (Triton/XLA/TVM) and on-the-fly autotuning; interconnect-aware graph lowering.
- Serving innovations (speculative decoding, distillation-on-demand).

OPS growth priors (average, 2026–2035 → 2036–2045):

- **CAGR:** 40–60%/yr → 25–40%/yr
- **Mechanism:** more capability per FLOP; slower OPS growth despite usage expansion.

8. Energy, Cost, and Carbon Sketch

8.1 From OPS to energy via performance/W assumptions

Let

- OPS(y) = average global AI operations per second (FLOP/s) in year y (from our series),
- η = arithmetic efficiency in FLOP/W (FLOP per second per watt, $\text{FLOP}\cdot s^{-1}\cdot W^{-1}$), expressed as a *FLOP-equivalent* rate for the prevailing precision/mix,
- PUE = power usage effectiveness (site power ÷ IT power),
- H = hours/year = 8760.

Then

- **IT power (W)**: P_IT = OPS / η
- **Site power (W)**: P_site = PUE * P_IT
- **Annual energy (kWh/year)**: E_kWh = (P_site * H) / 1000

If you prefer per-FLOP energy:

- **Joules per FLOP**: ε = 1 / η (J/FLOP)
- **kWh per FLOP**: ε_kWh = ε / 3.6e6 (since 1 kWh = 3.6 MJ)

Cost and carbon (with electricity price π in $/kWh and grid intensity γ in $kgCO_2/kWh$):

- **Annual cost ($/yr)**: Cost = E_kWh * π
- **Annual emissions (tCO_2/yr)**: CO2_t = (E_kWh * γ) / 1000

Note. Our OPS is already an *average* over the year; if you instead start from installed peak capacity, insert a separate utilization factor U and replace OPS with U * OPS_peak.

8.2 Sensitivities: PUE, duty cycle, inference mix

- **PUE (infrastructure overhead).** Typical envelopes today: **1.1–1.6**.
  Sensitivity: ∂ln E / ∂ln PUE = 1. A move from 1.4 → 1.2 trims site energy ~14%.
- **Arithmetic efficiency η (FLOP/W).** Driven by architecture, process, interconnect, memory hierarchy, precision (FP16/8/INT8/4), sparsity, and compiler fusion. Efficiency often varies by **×2–×5** across generations/settings.
  Sensitivity: ∂ln E / ∂ln η = −1. Doubling η halves energy for the same OPS.
- **Duty cycle / utilization U.** If starting from capacity, OPS_avg = U * OPS_peak. Queueing, bin-packing, preemption, and serving optimizers push **U** up; energy scales linearly with realized OPS.
- **Inference mix (and serving tricks).** High inference share can *raise* OPS (24/7 traffic) yet *cut per-request FLOPs* via quantization, sparsity, KV-cache reuse, speculative decoding, distillation. Net effect is model/stack dependent—treat as a scenario knob that acts through **η** and **OPS** jointly.
- **Rebound effects.** Better η and lower $/token often **increase total demand**, so do not assume energy drops with η—test both “efficiency-led with rebound” and “true savings” cases.

8.3 Back-of-envelope ranges and uncertainties

Below is a compact template you can use with *any* annual OPS from our series:

Inputs (pick within plausible bands)

- OPS_y: average global OPS for year y (FLOP/s).
- η: FLOP/W (FLOP-equivalent). Illustrative bands for mixed-precision AI fleets: **5e10–5e11 FLOP/W** (50–500 GFLOP/W).
- PUE: **1.1–1.6**.
- π ($/kWh): **0.04–0.20** (location/contract dependent).
- γ ($kgCO_2$/kWh): **0–0.8** (clean PPAs ↔ fossil-heavy grids).

Compute

```
P_IT   = OPS_y / η                    # W
P_site = PUE * P_IT                   # W
E_kWh  = (P_site * 8760) / 1000       # kWh/yr
Cost   = E_kWh * π                    # $/yr
CO2_t  = (E_kWh * γ) / 1000           # metric tons CO2/yr
```

Reading the levers

- **Order-of-magnitude checks.** If you scale OPS_y up by 10× (e.g., 2025 → 2030 demand-pull midline), **energy, cost, and CO_2 all rise ~10×** absent η/PUE changes.
- **η dominates.** A 3× η improvement offsets a 3× OPS increase; if both increase, energy holds flat.
- **PUE trims but doesn't dominate.** Going 1.3 → 1.15 saves ~13%.
- **Carbon depends on siting.** The same E_kWh yields **0–0.8 tCO_2/MWh**; clean PPAs/locations can reduce CO2_t by >5×.

Uncertainties to state explicitly

1. **OPS anchoring.** Our OPS is a transparent construction; substituting measured anchors (AI-DC power, accelerator counts × perf/W × U) will shift levels.
2. **η heterogeneity.** Fleet-average η blends generations, precisions, sparsity, interconnect penalties—expect ±(2–3×) variance.

3. **Hidden capacity / sovereigns.** Under-reported private or sovereign clusters may bias OPS_y up or down.
4. **Workload drift.** Agentic patterns and streaming inference can change both OPS_y and η in ways not captured by past trends.
5. **Rebound & policy.** Efficiency gains may be outpaced by demand; carbon hinges on PPAs, siting, and grid transitions.

9. Uncertainties and Limitations

9.1 Hidden capacity, under-reported private/sovereign clusters

Risk. Our usage-level OPS may be biased if large footprints are deliberately undisclosed (defense, sovereign labs, Tier-2 cloud/on-prem providers, hardware pilots under NDA) or if telemetry undercounts edge/embedded deployments. Bursty "black projects" (surge training, covert inference) can also distort annual averages.

What this does to the curve.

- **Level bias:** Shifts the whole OPS series up/down.
- **Kink risk:** Late-arriving disclosures can create artificial inflection points.
- **Regional skew:** Over/under-count varies by jurisdiction and export controls.

Mitigations.

- Publish ranges (low/central/high) and an **additive hidden-capacity prior** (e.g., +x% level shift with wide CI).
- Triangulate with **AI-DC power**, **HBM/package shipments**, and **accelerator install base × util × perf/W**.
- Version parameters so later revelations change a **release tag**, not the method.

9.2 Mixed precision equivalence; workload diversity

Risk. We normalize all arithmetic to FLOP-equivalents, but real fleets mix FP32/16/8, INT8/4, sparsity, MoE routing, KV-cache reuse, and fused operators. Different model classes (vision, LLMs, recsys, speech, RL) have **divergent FLOP footprints** per unit quality/latency. The mapping "work delivered → FLOPs" is non-stationary.

What this does to the curve.

- **Comparability drift:** A 1 TFLOP in 2016 may not be "the same work" as 1 TFLOP in 2025.
- **App-mix sensitivity:** Shifts in training:inference ratios or service mixes (e.g., agentic, streaming) alter realized OPS at constant capability.
- **Efficiency ambiguity:** Algorithmic gains show up as **fewer FLOPs per outcome**, potentially flattening OPS even as capability rises.

Mitigations.

- Keep FLOP-equivalents as the **primary yardstick**, but report companion **intensity ratios** (e.g., tokens/J, images/J) when available.
- Provide **scenario knobs** for precision mix, sparsity, and caching; show OPS results as bands rather than single lines.
- Document the **workload taxonomy** used (train vs infer, batch vs interactive) and update annually.

9.3 Model risk: era boundaries and structural breaks

Risk. Our piecewise-exponential assumption encodes eras (1995–2005, 2005–2012, 2012–2018, 2018–2025) and tapered projections thereafter. Reality may feature **structural breaks**: supply shocks (HBM, packaging), regulatory waves, radical efficiency jumps (e.g., stable 3–5× algorithmic savings), or demand shocks (agent ubiquity).

What this does to the curve.

- **Slope error:** True CAGR differs from assumed per-era growth.
- **Breakpoint misplacement:** Apparent "inflections" reflect assumption boundaries, not physics/economics.
- **Projection fragility:** Long-range paths (to 2045) accumulate error from compounding growth and policy/market feedbacks.

Mitigations.

- Report the tidy fit **with confidence bands** on log10(OPS) and publish **era-specific fits** side-by-side.
- Re-estimate annually using **robust/era-weighted regression**; test for breaks (e.g., Chow tests) when anchors change.

- Maintain **alternative projections**: supply-constrained, demand-pull, efficiency-led (Section 7) with explicit growth priors and doubling-time summaries.
- Separate **level** and **slope** uncertainty: show an intercept band (α) and a slope band (β/k) so readers can reason about each independently.

Practical reporting guidance.

- Always publish (i) assumptions file, (ii) code, (iii) CSV, (iv) figure hashes.
- State at least three uncertainties that dominate your current revision (e.g., hidden capacity ±x%, η variance ±y×, projection slope ±z%/yr).
- Prefer **interval claims** (e.g., "2025 average OPS is 2–6×10^14 FLOP/s") to point claims when anchors are sparse.
- Treat OPS as a **living statistic**: a reproducible baseline that is updated—not a hidden-method leaderboard.

10. Reproducibility and Update Plan

10.1 Files: CSV, plotting code, and regeneration steps

Artifacts (minimal set)

- data/global_ai_ops_estimate_1995_2045.csv — canonical year table
  Columns: year, avg_ops_per_sec_FLOPs, total_ops_in_year, cumulative_ops
- params/ops_model.v1.json — assumptions (baseline, era CAGRs, projection CAGRs, y0)
- src/build_ops_series.py — pure-Python generator (reads params, writes CSV, prints fit)
- notebooks/reproduce.ipynb — optional, mirrors the scripts for interactive users
- MANIFEST.md — lists files + SHA256 hashes (provenance)
- CHANGELOG.md — semantic-versioned changes (inputs, code, outputs)
- LICENSE — permissive license (e.g., CC BY 4.0 for figures/CSV, MIT for code)
- README.md — one-page "how to run, how to swap anchors"

One-command regeneration (Python)

fresh venv is encouraged; pinned reqs in requirements.txt (matplotlib, pandas, numpy only)

python src/build_ops_series.py --params params/ops_model.v1.json --out data/global_ai_ops_estimate_1995_2045.csv

python src/plot_ops.py --csv data/global_ai_ops_estimate_1995_2045.csv --outfig out/

Hashes (example)

Linux/macOS

shasum -a 256 data/global_ai_ops_estimate_1995_2045.csv > MANIFEST.md

PowerShell

Get-FileHash data\global_ai_ops_estimate_1995_2045.csv -Algorithm SHA256

Figures produced

- out/fig1_avg_ops_logfits.png (insert in §4.1)
- out/fig2_cumulative_ops.png (optional §4.2)
- out/fig3_scenario_bands.png (optional §7.4)

10.2 How to swap anchors (GPU shipments, AI power, capex)

Goal. Replace (or complement) the era-CAGR priors with measurable anchors. Three common anchor families:

A) Accelerator shipments / installed base

Inputs (per year):

- N_accel: new units shipped or installed
- Perf_accel: FLOP/s per accelerator (FLOP-equivalent at prevailing precision)
- U: utilization (0..1), average
- Optional: fleet attrition/retirements (stock-flow)

Mapping:

OPS_avg ~= (Installed_Accelerators * Perf_accel * U)

Installed_Accelerators_t = Installed_Accelerators_{t-1} * (1 - retire_rate) + N_accel_t

Where needed, split train vs infer fleets with different U.

B) AI-datacenter power (preferred when available)

Inputs (per year):

- P_site (MW): site power attributable to AI
- PUE: site/IT ratio
- eta (FLOP/W): arithmetic efficiency (fleet-average)

Mapping:

P_IT = P_site / PUE

OPS_avg ~= P_IT * eta

If only energy E_kWh is known: convert to average power and proceed.

C) Capex model (coarser)

Inputs (per year):

- AI_capex ($)
- Capex_to_OPS (FLOP/s per $ of capex), derived from BoM assumptions: accelerator $ share, interconnect, HBM, power/cooling, build times

Mapping (very approximate):

OPS_capacity_added ~= AI_capex * Capex_to_OPS

OPS_avg ~= f(capacity, ramp_profile, U)

Integration pattern

- Put anchors in params/anchors.v1.json and choose a **blend**:
- OPS_blend = w_era * OPS_era_model + w_anchor * OPS_anchor_model

with w_era + w_anchor = 1. Start with w_anchor = 0.3..0.5 where anchors are sparse.

Parameter file sketch (copy-safe JSON)

```
{
  "baseline": { "year0": 1995, "ops0_FLOPs_per_s": 1e9, "y0_center": 2020 },
  "eras": [
```

```
  {"start": 1995, "end": 2005, "cagr": 0.35},
  {"start": 2005, "end": 2012, "cagr": 0.50},
  {"start": 2012, "end": 2018, "cagr": 0.80},
  {"start": 2018, "end": 2025, "cagr": 1.00}
],
"projection": [
  {"start": 2026, "end": 2035, "cagr": 0.70},
  {"start": 2036, "end": 2045, "cagr": 0.40}
],
"anchors": {
  "use": true,
  "weight_anchor": 0.4,
  "ai_power_MW": { "2023": 8000, "2024": 12000 },  // example placeholder values
  "pue": 1.25,
  "eta_FLOPs_per_W": 1.5e11,
  "accelerators": {
    "shipments": { "2023": 1200000 },
    "perf_FLOPs_per_s": 2e14,
    "utilization": 0.45,
    "retire_rate": 0.08
  }
},
"scenario_bands": {
  "supply_constrained": {"g1_2026_2035": [0.35, 0.50], "g2_2036_2045": [0.20, 0.30]},
  "demand_pull":      {"g1_2026_2035": [0.90, 1.10], "g2_2036_2045": [0.60, 0.70]},
  "efficiency_led":   {"g1_2026_2035": [0.40, 0.60], "g2_2036_2045": [0.25, 0.40]}
```

}

}

Swap procedure (documented in README)

1. Drop new anchors into params/anchors.<date>.json.
2. Set anchors.use = true and adjust weight_anchor.
3. Run build_ops_series.py (it recomputes the series and recomputes the tidy fit).
4. Replot and refresh hashes; append to CHANGELOG.md.

10.3 Versioning and community pull-requests

Versioning

- **SemVer for the model**: MAJOR.MINOR.PATCH
    - MAJOR: breaking changes to definitions or units (e.g., OPS definition);
    - MINOR: new anchors, scenario changes, new plots (results shift meaningfully);
    - PATCH: bug fixes, typos, cosmetic figure tweaks (no methodological shift).
- Tag every release: v1.0.0, v1.1.0, etc. Upload CSV, figs, and params alongside the tag.

Changelog discipline

- For each version: list (a) changed inputs, (b) changed code, (c) changed outputs with deltas (e.g., "2025 OPS +18% vs v1.0").
- Include hashes of CSV and figures.

Review and tests

- **Unit tests**: deterministic regeneration from params; checks that totals and cumulative are monotone; reproducible fits (alpha, beta, k) within tolerances.
- **Golden files**: keep a small comparison CSV (5–10 years) to detect unintended drift.
- **CI**: on PR, auto-run build + plots, attach artifacts, and post diffs.

PR template (essentials)

- Motivation: anchors or method change
- Files touched: params/*.json, src/*.py, data/*.csv

- Evidence: before/after plots, key numbers (OPS in {2025, 2030, 2035})
- Risk: level vs slope effects; which sections of the paper change (e.g., §4.1 paragraph 2)

Governance

- Keep **definitions and scope** (§2) as a protected doc; changes require MAJOR bump.
- Require at least one reviewer to sign off that **units** and **counting rules** remain intact (FLOP-equivalent arithmetic only; no IO).

Attribution & reuse

- Encourage forks with different priors (e.g., sovereign capacity adjustments). Require forks to:
    - keep method code intact or clearly flagged,
    - publish params + anchors,
    - publish a diff table vs upstream (OPS deltas by year).

Data ethics

- Avoid ingesting non-public operator leaks. If a public source is later retracted, tag a corrective MINOR release with the anchor removed and note the effect on the curve.

Bottom line: anyone should be able to 1) clone, 2) edit a single JSON, 3) run two scripts, 4) reproduce the CSV and figures, and 5) verify them with hashes. That keeps the OPS baseline living, transparent, and easy to re-anchor as better measurements arrive.

11. Implications and Use Cases

11.1 Strategy: capacity planning, chip roadmaps, power siting

Capacity planning. Treat OPS as the first-class demand signal. Translate target service levels (tokens/s, images/s, agent-steps/s) into annual OPS, then back-solve for: (i) accelerators (count × perf), (ii) interconnect bandwidth, (iii) HBM/package lanes, and (iv) siting power. Maintain a rolling **OPS budget** with three ledgers: training, inference, and “experiments” (R&D spikes).
Chip & system roadmaps. Align architecture bets to the OPS mix you expect (e.g., high KV-cache reuse and low-bit inference → memory- and network-biased designs). For training cadence, quantify how much OPS you need to refresh models (N train runs/year × FLOPs/run). Publish **target perf/W** and **effective FLOPs/token** goals per generation.

Power siting. Convert OPS targets to **IT MW** via fleet η (FLOP/W) and add PUE. Pre-commit grid capacity with: (a) multi-year PPAs, (b) transmission upgrades lead-time, (c) waste-heat offtake plans. Use **OPS/MW** as a governance KPI: rising OPS/MW signals real efficiency progress.
Resilience. Stress-test OPS continuity under HBM/co-packaging shortages, node slips, or curtailments. Keep **burst OPS** reserves (idle headroom or "interruptible" agents) to absorb demand spikes without violating SLOs.

11.2 Governance: reporting norms, compute disclosures

Standard disclosures. Once per year (or quarter for large operators) report:

- **Average OPS** (FLOP/s), **total annual FLOPs**, and **cumulative FLOPs**;
- **Train:infer share** (by OPS), **precision mix** (FP16/8/INT8/4), and **utilization** U;
- **Perf/W (η)** at fleet-average; **PUE**; **site MWh**; **grid mix** or PPA shares.
  Interpretability. Always provide the **mapping** from service metrics (tokens, images, requests) to OPS (e.g., FLOPs/token, FLOPs/image), and publish method notes for compression, sparsity, caching.
  Thresholding & safety. Tie capability governance to **usage gates**, not device counts: e.g., external review above X×10^23 FLOPs/train or Y×10^20 FLOPs/day inference. Couple with **energy/carbon gates** (tCO_2 per 10^20 FLOPs) and **water** disclosures where relevant.
  Auditability. Release **parameter files** (anchors, η, PUE) and hash-stamped CSVs. Allow third-party recomputation from disclosed anchors. Encourage sector consortia (clouds, OEMs, and labs) to adopt comparable templates so regulators can build OPS-level macro-ledgers.

11.3 Research: benchmarking OPS vs outcomes

Efficacy per FLOP. Establish benchmarks that report **quality per FLOP** (e.g., accuracy/bleu/safety metrics per 10^x FLOPs) alongside classic scores. Track **frontier efficiency** (best-known outcomes per FLOP) and **production efficiency** (fleet-average outcomes per FLOP).
Elasticity studies. Quantify how outcomes scale with OPS across tasks (vision, language, speech, RL). Separate **algorithmic efficiency** (fewer FLOPs per outcome) from **scale returns** (better outcome per more FLOPs). Publish **learning-curve exponents** with FLOPs on the x-axis.
Serving science. For inference, measure **FLOPs/request** under real SLOs, with/without speculative decoding, caching, MoE, and distillation. Report **tokens/J** and **requests/J** as paired metrics.
Systems knobs. Run controlled studies varying precision (FP16→FP8→INT4), sparsity, interconnect topologies, and compiler stacks; publish η shifts and their impact on wall-clock OPS and quality.

Public corpora & receipts. Release **OPS-annotated** training recipes (FLOPs by phase: pretrain, SFT, RL, eval) and anonymized **inference receipts** (distribution of FLOPs/request). Standardize a lightweight **opslog.json** for paper artifacts so others can reproduce OPS tallies.
Policy-relevant syntheses. Tie OPS to **externalities** (MWh, tCO_2, kGal water) and **societal outputs** (productivity measures, error rates, safety incidents). Provide **counterfactuals**: what OPS would be required to achieve target outcome deltas (e.g., +2 accuracy points) under different efficiency assumptions.

Bottom line. Treat OPS as a shared accounting layer—one number that connects silicon, software, workloads, and societal cost. Planning, governance, and science all become more legible when results are expressed per unit of arithmetic actually performed.

12. Conclusion

12.1 What the OPS lens clarifies

The OPS lens (average FLOP/s actually *used*) turns a fragmented conversation—chips, parameters, capex, anecdotes—into a single, auditable quantity that spans training *and* inference across cloud, on-prem, and edge. It exposes real growth dynamics (capacity × utilization × software efficiency), makes era shifts legible (pre-GPU → DL boom → transformer surge), and allows apples-to-apples comparisons with Moore-family trends. Because OPS integrates both supply (hardware, siting, power) and demand (workload mix, SLOs), it becomes a natural base for scenario bands, energy/carbon accounting, and governance thresholds. In short, OPS is a practical lingua franca linking silicon, systems, economics, and policy.

12.2 What it can't resolve (yet)

OPS does not, by itself, certify *capability* or *value*. One FLOP in 2015 is not the same "work" as one FLOP in 2025; algorithmic gains and precision changes shift the FLOP-to-quality conversion. Hidden capacity (sovereigns, private clusters), ambiguous workload boundaries, and uneven disclosures introduce level and slope uncertainty. OPS also cannot decide normative questions—how much AI the world *should* run, or which uses deserve priority. Finally, long-horizon projections are fragile: structural breaks (packaging, HBM, power, regulation, agentic demand) can bend the curve faster than any extrapolation admits.

12.3 A living baseline for the next decade

The remedy is process, not perfection. We propose a living baseline: a tiny, open toolchain that regenerates the OPS series from explicit parameters, welcomes anchor swaps (AI power, accelerator stock, utilization studies), and publishes hashes, figures, and changelogs. Annual re-estimation with era-aware fits and scenario envelopes keeps the narrative honest while

preserving comparability. Pair OPS with companion intensities (tokens/J, requests/J) so capability and efficiency co-evolve on the record. If the community treats OPS like national accounts for AI—transparent, versioned, and reproducible—we can argue less about proxies and more about outcomes: where to invest, what to govern, and how to align arithmetic with human aims.

Appendices

A. Equations and parameter tables (alpha, beta, A, k, doubling times)

A.1 Canonical "tidy" fit (historical 1995–2025, centered at y0 = 2020)

- Centered log form
  log10(OPS(y)) = alpha + beta * (y - y0)
  alpha = 13.575, beta = 0.203324, y0 = 2020
- Natural-exp form (equivalent)
  OPS(y) = A * exp(k * (y - y0))
  A = 3.766935e13 FLOP/s (level at y = 2020)
  k = beta * ln(10) = 0.468171 / year
- Growth summaries
  Doubling time T2 = ln(2) / k = 1.48 years
  CAGR ~= 10^beta - 1 ~= 0.597 (~59.7 %/yr)
  Goodness of fit (1995–2025), R^2 ~= 0.972 (log10 space)
- Quick inversion
  Given target OPS*, year y = y0 + (1/k) * ln(OPS* / A)

A.2 Era-specific tidy fits (constructed series; centered per era)

Era	y0 (center)	A at y0 (FLOP/s)	k (/yr)	Doubling T2 (yr)	CAGR (/yr)	log10 slope beta	R^2
1995–2005 (pre-GPU)	2000.0	4.484033e9	0.300105	2.3097	0.3500	0.130334	~1.000
2005–2012 (GPU/DC)	2008.5	8.311073e10	0.405465	1.7095	0.5000	0.176091	~1.000
2012–2018 (DL boom)	2015.0	2.003522e12	0.587787	1.1793	0.8000	0.255273	~1.000
2018–2025 (xformers)	2021.5	1.370057e14	0.714705	0.9698	1.0436	0.310393	~0.999

(Values reflect the constructed usage series; re-anchoring will shift A and possibly k.)

B. Cumulative closed-form and discrete comparison

B.1 Definitions

- Annual average OPS: OPS(y) in FLOP/s
- Seconds per year: S = 31,557,600
- Annual total FLOPs: YearFLOPs(y) = OPS(y) * S
- Discrete cumulative through year N: Cum_disc(N) = sum_{y=1995}^{N} YearFLOPs(y)

B.2 Continuous closed-form (using the tidy exponential)

If OPS(y) = A * exp(k * (y - y0)), cumulative from y_start to y is:

Cum_cont(y) = S * (A / k) * [exp(k * (y - y0)) - exp(k * (y_start - y0))]

Use y_start = 1995 for this paper.

B.3 How to compare (recommended in repo)

1. Compute Cum_disc(y) from the CSV.
2. Compute Cum_cont(y) from the tidy fit.
3. Report log10 residuals and R^2 on y >= 1996 (avoid log10(0) at the baseline).
4. Expect tight agreement for a single-slope era; mild, structured deviations around era transitions.

C. Scenario parameter ranges and example calculations

C.1 Two-stage CAGR scaffolds (used for Fig. 3)

Let OPS_2025 be the 2025 anchor (~3.9e14 FLOP/s from the tidy fit). For t in years, propagate:

OPS_{t+1} = OPS_t * (1 + g_t)

with stagewise g_t picked from the ranges below.

- Supply-constrained
  2026–2035: g in [0.35, 0.50] (mid = 0.425)
  2036–2045: g in [0.20, 0.30] (mid = 0.25)

- Demand-pull
  2026–2035: g in [0.90, 1.10] (mid = 1.00)
  2036–2045: g in [0.60, 0.70] (mid = 0.65)

- Efficiency-led
  2026–2035: g in [0.40, 0.60] (mid = 0.50)
  2036–2045: g in [0.25, 0.40] (mid = 0.325)

C.2 Illustrative midline snapshots (order-of-magnitude)

Using OPS_2025 ~= 3.9e14 FLOP/s:

- Supply-constrained (mid: 42.5%/yr then 25%/yr)
  2030: factor (1.425)^5 ~= 5.9 ⇒ ~2.3e15 FLOP/s
  2035: factor (1.425)^10 ~= 34.5 ⇒ ~1.35e16 FLOP/s
  2040: *then* (1.25)^5 ~= 3.05 ⇒ ~4.1e16 FLOP/s
  2045: *then* ×3.05 again ⇒ ~1.3e17 FLOP/s

- Demand-pull (mid: 100%/yr then 65%/yr)
  2030: 2^5 = 32 ⇒ ~1.25e16 FLOP/s
  2035: 2^10 = 1024 ⇒ ~4.0e17 FLOP/s
  2040: (1.65)^5 ~= 12.2 ⇒ ~4.9e18 FLOP/s
  2045: ×12.2 again ⇒ ~6.0e19 FLOP/s

- Efficiency-led (mid: 50%/yr then 32.5%/yr)
  2030: 1.5^5 ~= 7.6 ⇒ ~3.0e15 FLOP/s
  2035: 1.5^10 ~= 57.7 ⇒ ~2.2e16 FLOP/s
  2040: (1.325)^5 ~= 4.1 ⇒ ~9.0e16 FLOP/s
  2045: ×4.1 again ⇒ ~3.8e17 FLOP/s

These are illustrative midlines, not forecasts; swap in your preferred anchors and re-emit Fig. 3.

C.3 Energy sketch tie-in (one-liner)

Given fleet eta (FLOP/W) and PUE:

P_IT = OPS / eta; P_site = PUE * P_IT; E_kWh = (P_site * 8760) / 1000

Use this with the OPS points above to tabulate site MWh, cost, and CO2 per scenario.

D. Data/Code manifest and checksum notes

D.1 Minimal manifest (paper repo)

```
/data
  global_ai_ops_estimate_1995_2045.csv

/params
  ops_model.v1.json          # baseline, era/projection CAGRs, y0
  anchors.v1.json            # optional: ai_power, pue, eta, accel stock/flow

/src
  build_ops_series.py        # generates CSV from params (+ tidy fit, Table 1)
  plot_ops.py                # Fig.1 (log-fits), Fig.2 (cumulative), Fig.3 (bands)

/out
  fig1_avg_ops_logfits.png
  fig2_cumulative_ops.png
  fig3_scenario_bands.png

MANIFEST.md                  # filenames + SHA256
CHANGELOG.md                 # semver and deltas
README.md                    # run steps, how to swap anchors
LICENSE
```

D.2 Checksums (example commands)

- Linux/macOS
  shasum -a 256 data/global_ai_ops_estimate_1995_2045.csv >> MANIFEST.md
- Windows PowerShell
  Get-FileHash data\global_ai_ops_estimate_1995_2045.csv -Algorithm SHA256

Include hashes for CSV and all figures in MANIFEST.md. When you regenerate, append a new section with date/time and updated hashes.

D.3 Regeneration recipe (single pass)

python src/build_ops_series.py --params params/ops_model.v1.json --out data/global_ai_ops_estimate_1995_2045.csv

python src/plot_ops.py --csv data/global_ai_ops_estimate_1995_2045.csv --outfig out/

D.4 Versioning notes

- SemVer the *method* (MAJOR.MINOR.PATCH).
- Tag releases and archive the CSV + figures at each tag.
- In CHANGELOG, report deltas (e.g., “OPS(2025) +18% vs v1.0 due to updated AI-DC power anchor”).
- Keep /params immutable by version (e.g., ops_model.v1.json, ops_model.v2.json) to preserve provenance.

References

Barroso, L. A., & Hölzle, U. (2007). The case for energy-proportional computing. *IEEE Computer*, 40(12), 33–37.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). cuDNN: Efficient primitives for deep learning. *arXiv:1410.0759*.

Dennard, R. H., Gaensslen, F. H., Yu, H.-N., Rideout, V. L., Bassous, E., & LeBlanc, A. R. (1974). Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5), 256–268.

Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). LLM.int8(): 8-bit matrix multiplication for transformers at scale. *NeurIPS*.

Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. *ICLR*.

Hennessy, J. L., & Patterson, D. A. (2019). *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann.

Hernandez, D., & Brown, T. (2020). Measuring the algorithmic efficiency of neural networks. *arXiv:2005.04305*.

Hoffmann, J., Borgeaud, S., Mensch, A., et al. (2022). Training compute-optimal large language models. *arXiv:2203.15556* ("Chinchilla").

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., … Adam, H. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR*.

Jouppi, N. P., Young, C., Patil, N., et al. (2017). In-datacenter performance analysis of a tensor processing unit. *ISCA*.

Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). Scaling laws for neural language models. *arXiv:2001.08361*.

Koomey, J. G. (2011). Growth in data center electricity use 2005 to 2010. *Analytics Press*.

Koomey, J. G., Berard, S., Sanchez, M., & Wong, H. (2011). Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing*, 33(3), 46–54.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *NeurIPS*.

Kwon, W., Lee, S., Li, Z., et al. (2023). Efficient memory management for large language model serving with PagedAttention. *SOSP* (vLLM).

Leary, C., & Wang, T. (2017). XLA: TensorFlow, compiled. *TensorFlow Dev Summit* (whitepaper/talk).

Masanet, E., Shehabi, A., Lei, N., Smith, S., & Koomey, J. (2020). Recalibrating global data center energy-use estimates. *Science*, 367(6481), 984–986.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8), 114–117.

Moore, G. E. (1975). Progress in digital integrated electronics. *IEDM Technical Digest*, 11–13.

Shazeer, N., Mirhoseini, A., Maziarz, K., et al. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*.

Tillet, P., Cox, J., & Kung, H. T. (2019). Triton: An intermediate language and compiler for tiled neural network computations. *ICML Systems for ML Workshop* (and subsequent docs).

The Green Grid. (2007). The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE. *White Paper No. 6*.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. *NeurIPS*.

Wei, J., Chen, S., Zhou, Y., et al. (2023). Speculative decoding for accelerated generation in large language models. *arXiv:2302.01318*.

Note: Citations to software stacks (cuDNN, XLA, Triton) and operational metrics (PUE) reference canonical whitepapers or widely cited introductions. Where arXiv identifiers are given, use the most recent stable versions available when finalizing the bibliography.