

The Half-Life of Compute Thresholds*

Aaron Potter
Independent Researcher

With
Apart Research

Abstract

Compute thresholds are widely discussed as a practical governance tool, but the literature rarely specifies how quickly fixed thresholds become outdated as algorithmic efficiency improves. We develop a compute-threshold staleness model that distinguishes raw training compute from baseline-equivalent (“effective”) compute and defines staleness as the ratio between a policy threshold and the time-varying threshold that would remain aligned with a fixed risk cutoff. Under exponential efficiency gains with doubling time τ and threshold updates every U months, worst-case staleness grows as $S_{\max} = 2^{U/\tau}$, implying that modest uncertainty about τ can translate into large differences in required update cadence. Using a representative frontier estimate of $\tau \approx 8$ months suggests that annual recalibration can allow thresholds to drift by roughly $2.8\times$ within one year. We ground plausible values of τ in prior estimates of algorithmic progress and provide a small historical backtest as a sanity check.

1 Introduction

Compute thresholds are widely used in regulatory frameworks for AI models (Pistillo et al., 2025), as they correlate roughly with risk and are relatively easy to predict ahead of time. The literature also observes that compute thresholds will need to be shifted (ArXiv discussion, 2024). How much, how frequently, and in what direction lacks any clear consensus. This paper models the impact of algorithmic progress on how quickly thresholds become stale, an important consideration especially for handling CBRN risks, which unlike loss-of-control risks are not clustered at the frontier.

Our main contributions are:

1. A simple mathematical model applying the observed pattern of exponential growth in algorithmic efficiency to questions of staleness of compute thresholds
2. A dashboard to model the relationship between the update frequency, the doubling time of algorithmic efficiency, and the staleness of compute thresholds

*Research conducted at the Technical AI Governance Challenge, 2026.

2 Related Work

Compute thresholds in AI regulation

Training Compute Thresholds: Features and Functions in AI Regulation. The analysis of compute thresholds in this paper is heavily indebted to Heim and Koessler’s discussion (Heim and Koessler, 2024). Their work is primarily qualitative, and we use it mainly to motivate why thresholds are appealing as an initial filter and why update mechanisms matter. In particular, they discuss the suitability of compute thresholds for an initial filter, the need for updates, and tradeoffs in both conservative and aggressive thresholds, as well as how compute thresholds are used in existing policy (principally the EU AI Act).

Algorithmic progress and efficiency trends

Algorithmic progress in language models. This paper provides an analysis and rough estimate of the role of algorithmic progress in making training more efficient to reach a given capability level (Ho et al., 2024). We use their empirical estimates to motivate plausible values of τ (an “effective compute” doubling time) in our staleness model; as shown in Section 4, the implications are highly sensitive to τ .

Positioning and gap

Despite the existence of extensive work on algorithmic progress, and the adoption of compute thresholds with the acknowledgment of a need to update them, there is little work on modeling the relationship of update frequency and algorithmic progress to the degree with which a model falls out of date. This paper attempts to close that gap with a simple model which can provide a good intuitive grasp of the problem for policy-makers.

3 Methods

3.1 Compute Threshold Staleness Model

We model time as $t \in \mathbb{R}_{\geq 0}$ (months, mo), with baseline month $t = 0$. The policy observes raw training compute $C > 0$ (FLOPs).

Efficiency and effective compute. Let $A(t)$ be a dimensionless algorithmic-efficiency multiplier with $A(0) = 1$, parameterized by a doubling time τ (months):

$$A(t) = 2^{t/\tau}. \tag{1}$$

Define effective compute (baseline-equivalent FLOPs)

$$E(C, t) := A(t) C. \tag{2}$$

Risk and policy trigger. A run is “risky” if effective compute exceeds a fixed cutoff $E_{\text{risk}} > 0$:

$$R := \{E(C, t) \geq E_{\text{risk}}\}. \quad (3)$$

A compute-threshold policy triggers when

$$G := \{C \geq T(t)\}, \quad T(t) > 0. \quad (4)$$

Ideal threshold and staleness. The ideal raw-compute threshold that exactly aligns with the risk boundary at month t is

$$T^*(t) = \frac{E_{\text{risk}}}{A(t)}. \quad (5)$$

Define threshold staleness as

$$\text{Staleness}(t) := \frac{T(t)}{T^*(t)} = \frac{A(t)T(t)}{E_{\text{risk}}} = \frac{E_{\text{th}}(t)}{E_{\text{risk}}}, \quad (6)$$

where $E_{\text{th}}(t) := A(t)T(t)$ is the effective compute at the policy threshold.

Step-hold updates every U months. Let $U > 0$ be the update interval and $t_n := nU$. Under a step-hold policy,

$$T(t) := T^*(t_n) = \frac{E_{\text{risk}}}{A(t_n)} \quad \text{for } t \in [t_n, t_{n+1}). \quad (7)$$

Within an interval, staleness depends only on efficiency growth since the last update:

$$\text{Staleness}(t) = \frac{T^*(t_n)}{T^*(t)} = \frac{A(t)}{A(t_n)} = 2^{(t-t_n)/\tau}. \quad (8)$$

Hence the worst-case staleness just before the next update is

$$\text{Staleness}_{\text{max}} = 2^{U/\tau}, \quad (9)$$

and the time to become $m \times$ stale within an interval is

$$t - t_n = \tau \log_2 m. \quad (10)$$

4 Results

This section reports closed-form implications of the staleness model and illustrates their sensitivity to the key ratio U/τ . We treat the historical comparisons as a lightweight sanity check; the primary empirical grounding for τ comes from Ho et al. (2024).

4.1 Model Sensitivity to Key Parameters

The staleness of a compute threshold is determined by the ratio U/τ , where U is the policy update interval (in months) and τ is the algorithmic efficiency doubling time (in months). The maximum staleness just before a policy update is given by:

$$S_{\max} = 2^{U/\tau} \quad (11)$$

This relationship is highly sensitive to τ . For example, with an annual update cycle ($U = 12$ months):

- At $\tau = 8$ months (frontier estimate from Ho et al. (2024)): $S_{\max} = 2^{1.5} \approx 2.8\times$
- At $\tau = 4$ months: $S_{\max} = 2^3 = 8\times$
- At $\tau = 16$ months: $S_{\max} = 2^{0.75} \approx 1.7\times$

The update interval required to maintain a target maximum staleness S_{target} is:

$$U = \tau \cdot \log_2(S_{\text{target}}) \quad (12)$$

4.2 Implications of 8-Month Doubling Time

Using the Ho et al. (2024) estimate of $\tau = 8$ months for frontier algorithmic efficiency improvements:

- The ideal compute threshold halves every 8 months
- A threshold set today will be approximately $2.8\times$ too permissive after one year without updates
- After two years, a static threshold becomes $8\times$ too permissive
- To maintain staleness $\leq 2\times$, updates are needed every $U = 8 \cdot \log_2(2) = 8$ months

These results suggest that annual update cycles—common in regulatory contexts—may be insufficient to maintain threshold calibration if algorithmic progress continues at frontier rates.

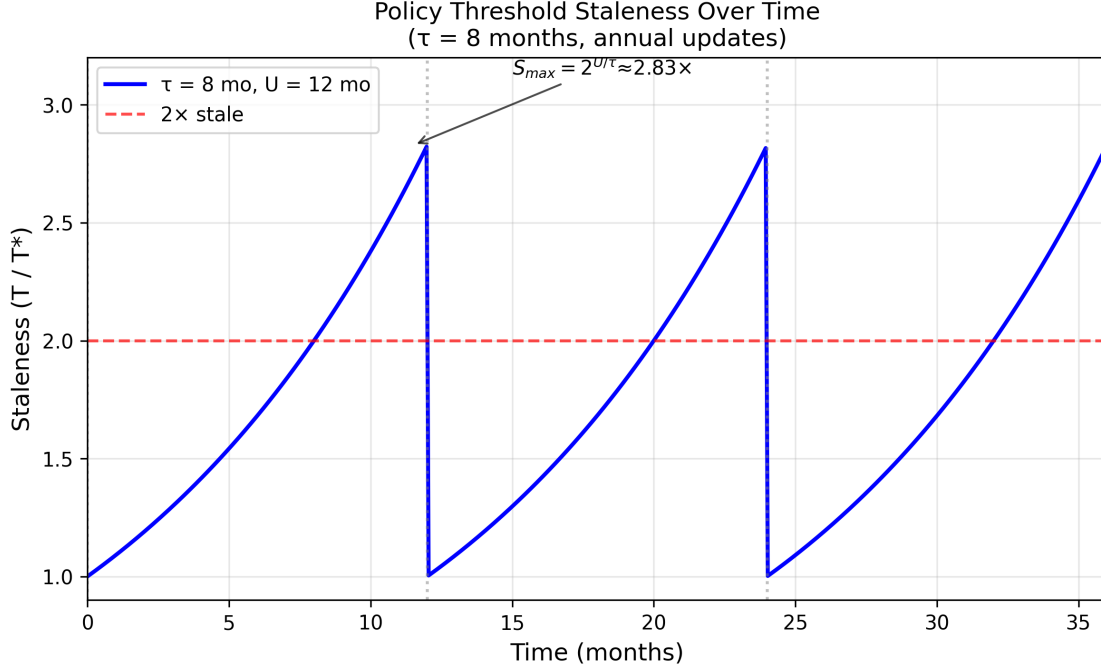


Figure 1: Staleness growth over time under an 8-month algorithmic-efficiency doubling time ($\tau = 8$ months).

4.3 Historical Validation

A backtest against historical AI model releases provides a rough sanity check of the τ parameter. We compare capability-matching model pairs—cases where a later, smaller model achieved similar benchmark performance as an earlier, larger model with less training compute.

Data Sources. Training compute estimates are sourced from Epoch AI’s Notable AI Models dataset (Epoch AI, 2024) where available, supplemented by the standard scaling approximation $C \approx 6ND$ (where N is parameter count and D is training tokens) for models with disclosed training configurations (Kaplan et al., 2020). This approximation is coarse and depends on training details that are not always publicly reported. Release dates are taken from official announcements or first public availability.

Capability Matching. We identify pairs where the matching model uses *less* compute than the reference while achieving comparable scores on specific benchmarks (e.g., HellaSwag, MMLU). Pairs where the later model used more compute are excluded, as these test data efficiency rather than algorithmic efficiency. The “comparable capability” judgment is approximate and benchmark-specific—a significant limitation of this validation approach.

Results. Given a compute ratio $r = C_{\text{match}}/C_{\text{ref}}$, the model predicts a time gap of $\Delta t = \tau \cdot \log_2(1/r)$ for capability parity. Minimizing RMSE across all pairs yields a best-fit $\tau \approx 9.5$ months, broadly

consistent with the frontier estimate of 8 months reported by Ho et al. (2024). This convergence is

Reference	Matching Model	Compute Ratio	Actual Gap	Predicted Gap*
GPT-3 175B	LLaMA-13B	$0.35\times$	32 mo	28 mo
GPT-3 175B	Mistral 7B	$0.27\times$	40 mo	34 mo
GPT-4	LLaMA 3 70B	$0.43\times$	13 mo	10 mo
GPT-4	Qwen 2.5 72B	$0.36\times$	18 mo	12 mo

Table 1: Capability-matching pairs used in backtest. Compute ratios indicate how much less compute the matching model used relative to the reference. *Predicted gap using $\tau = 8$ months.

expected given that the model’s efficiency dynamics rely primarily on Ho et al. (2024)’s empirical estimates of frontier algorithmic progress. The backtest confirms internal consistency rather than providing independent validation of τ .

5 Discussion and Limitations

These results indicate that, to keep compute thresholds well-calibrated, the update cadence should scale with the doubling time of algorithmic efficiency. However, τ is uncertain and may differ across regimes (e.g., frontier vs catch-up), making calibration difficult. This uncertainty is especially salient as models cross capability levels associated with significant CBRN risk. Where loss-of-control concerns may cluster near the frontier (which is hard to define but relatively easy to track), CBRN-relevant capabilities may not be confined to the frontier.

This suggests that improving estimates of algorithmic progress (i.e., of τ) is important for selecting an appropriate update cadence. Finally, if efficiency gains slow over time (e.g., a sigmoidal trajectory due to physical limits), threshold drift may eventually become less severe.

Limitations

This model is highly idealized and cannot give detailed prescriptions. For example, capabilities do not necessarily improve at the same rate across domains as algorithmic efficiency changes. The model also ignores error in setting new thresholds, which could be significant in practice. If the rate of algorithmic progress is continually underestimated (e.g., in a pause world where most or all algorithmic research is covert), then staleness could grow substantially.

More fundamentally, effective compute is a modeling construct rather than an observable quantity. There may be no single scalar that captures the capabilities we care about across domains and threat models. As a result, calibration will necessarily require judgment beyond estimating a single doubling time τ —including choices about which benchmarks, evaluations, and risk cutoffs the threshold is intended to track.

Future Work

This model could be expanded to consider other parameters, though high uncertainty makes detailed modeling difficult and somewhat prone to false precision risks. The biggest suggestion is that accurate research on algorithmic progress may be necessary to make compute thresholds workable; absent that, compute thresholds may not be reliable for identifying models likely to pose substantial risks.

6 Conclusion

This project formalizes a simple model for the degree to which a compute threshold would be expected to be out of date as a result of improvements in algorithmic efficiency. Applying the currently observed model of exponential growth of algorithmic efficiency, the model predicts that compute thresholds have worst-case staleness $S_{\max} = 2^{U/\tau}$. A small historical backtest suggests the model’s predictions are consistent with an 8–10 month regime, in line with Ho et al. (2024).

Applying a representative frontier estimate of $\tau \approx 8$ months implies that annual recalibration can allow thresholds to drift by roughly $2.8\times$ within a year, suggesting that annual updates may leave thresholds meaningfully miscalibrated for a substantial portion of the year. This model provides a compact way to reason about the relationship between thresholds and observed trends in algorithmic efficiency, with potential uses in policy planning and communication, though it is a highly ideal model and should not be used in place of empirical observation.

Code and Data

- Code repository: <https://github.com/Ardrual/Compute-Threshold-Staleness-Model>

References

- Matteo Pistillo, Suzanne Van Arsdale, et al. *The Role of Compute Thresholds for AI Governance*. *Geo. Wash. J. L. & Tech*, 1:26, 2025.
- Lennart Heim and Florian Koessler. *Training Compute Thresholds: Features and Functions in AI Regulation*. 2024.
- Eric Ho, et al. *Algorithmic progress in language models*. 2024.
- Epoch AI. *Notable AI Models*. 2024. <https://epochai.org/data/notable-ai-models>.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. *Scaling Laws for Neural Language Models*. 2020.
- Discussion of updating compute thresholds*. arXiv:2405.10799v2, Section 6. <https://arxiv.org/html/2405.10799v2#S6>.

A Appendix

LLM Usage Statement

The initial mathematical formalism was developed with substantial input from GPT-5.2, Claude Opus 4.5, and Gemini Pro 3. The python model was programmed primarily by GPT-5.2-Codex, while the frontend was primarily programmed by Claude Opus 4.5 and Gemini 3 Fast. The writeup was done with substantial formatting and editing help from GPT-5.2 in Prism.