Adaptive Recursive Dimensional Compression (ARDC): A Scaling Law for Emergent Efficiency

Jaysen S. Lamb
Valid Systems / Independent Researcher
jlamb@validadvantage.com

December 30, 2025

"Efficiency emerges when redundancy removal surpasses information decay."

Title — Recursive Engine — (R Δ)

Abstract

This paper introduces Adaptive Recursive Dimensional Compression (ARDC): a measurable efficiency relation for controlled recursion in learning and decision systems. ARDC models the tradeoff between redundancy removal and information decay by defining efficiency

$$E(n) = 2^{-(n-1)}\, M_n\, C_n^2 \left(\frac{K_n}{\mu_n}\right), \tag{1}$$

where n is recursion depth, M_n is informational mass (effective dimension), C_n is clarity (bounded coherence proxy), K_n is knowledge integrity (retained task-relevant performance), and μ_n is uncertainty load (strictly positive risk/entropy proxy). The key claim is operational: recursion should be audited and stopped when marginal efficiency gains are exhausted under integrity constraints, rather than pushed deeper by default. Empirical results and controller demonstrations show a consistent crossover behavior: when redundancy removal dominates information decay, recursion transitions from destructive compression to adaptive abstraction, yielding higher performance-per-cost. ARDC therefore acts as a falsifiable stopping law that links information structure, computational cost, and governed recursion depth.

Significance Statement

ARDC identifies a measurable balance point between redundancy removal and information decay: the threshold where compression stops destroying meaning and begins revealing structure. By tying recursion depth to auditable quantities—retained knowledge (K), clarity (C), informational mass (M), and uncertainty load (μ)—ARDC converts "depth" into a governed, testable control decision. The framework is designed to be falsifiable: at each depth, ARDC logs $(M_n, C_n, K_n, H(n), \mu_n, E(n))$ under a fixed evaluation protocol and selects an optimal depth n^* by maximizing measured efficiency subject to integrity constraints. In practice, ARDC does not reward recursion; it rewards stopping at the right depth.

1 Introduction

Modern machine-learning and financial systems struggle to scale under uncertainty. As models ingest higher-dimensional inputs, efficiency often collapses through overfitting, entropy accumulation, and failure to distinguish durable signal from noise. In response, practitioners often add parameters, compute, features, or engineering complexity—which can improve short-term scores while increasing long-run brittleness and operating cost.

Why ARDC is necessary. A common failure pattern in modern systems design is the assumption that "more"—layers, parameters, epochs, features, or recursion steps—is inherently better. Existing heuristics (early stopping, dropout, regularization schedules, ad hoc risk overrides) can be effective, but they are typically domain-tuned and reactive; they do not provide an auditable, quantitative rule for when additional recursion becomes waste or risk.

ARDC addresses this blind spot by defining an explicit efficiency law $E(n)$ grounded in measurable structure: retained signal K_n, clarity C_n, informational mass M_n, and uncertainty load μ_n. The core posture shifts from "how deep can we go?" to "does the next step improve retained knowledge per unit uncertainty and compute?"

Contributions and Scope

This paper contributes:

- A single governing ARDC efficiency relation with defined variables (Eq. (1)).
- An operational measurement protocol that makes claims testable and comparable across depth.
- A depth-selection controller that outputs a complete audit log and explicit stop reason.
- Falsifiable predictions and failure conditions.

- Cross-domain empirical framing (synthetic compression, real-world tabular, finance/backtest contexts).

ARDC is presented as an entropy-guided efficiency governor for recursion and compression in learning/decision systems, not a universal storage codec and not a claim of physical mass–energy equivalence.

2 Related Work and Positioning

Where ARDC sits. ARDC is not proposed as a new learning algorithm; it is a *governance-grade control rule* for deciding when recursion (compression, pruning, iterative refinement) is still paying for itself. It is therefore best compared to methods that regulate complexity (early stopping, MDL/AIC-style penalties, information bottleneck objectives, and compression-progress priors), rather than to any single model architecture.

2.1 Early Stopping vs. Efficiency Stopping

Early stopping typically monitors a single proxy (e.g., validation loss) and halts when that proxy stops improving. This is useful but incomplete: a system can maintain flat validation score while becoming *less efficient* due to rising uncertainty, instability, or cost floors. ARDC differs by stopping on *audited efficiency*:

$$E(n) = 2^{-(n-1)}\, M_n\, C_n^2 \left(\frac{K_n}{\mu_n}\right),$$

so recursion is continued only when the measured ratio of retained knowledge to uncertainty (and cost proxies) improves. In other words, early stopping is *score-first*; ARDC is *efficiency-first under integrity constraints.* This distinction matters in regulated settings where operational cost, stability, and risk amplification are first-class concerns.

2.2 Minimum Description Length and Penalized Complexity

The MDL principle frames learning as a tradeoff between fit and description length (complexity), where models that compress data well without overfitting are preferred (MacKay, 2003). ARDC is compatible with this posture but focuses on a different observable: instead of optimizing an explicit code length, ARDC audits recursion depth using task retention K_n, clarity C_n, and an uncertainty load μ_n. Practically, this makes ARDC deployable even when an explicit coding model is unavailable, while retaining the MDL spirit: complexity must be paid for by measurable generalizable structure.

2.3 Information Bottleneck and Controlled Representation

The Information Bottleneck (IB) method formalizes representation learning as a tradeoff between compressing inputs and preserving information relevant to targets (Tishby et al., 2000). ARDC can be read as an operational control companion to IB: K_n plays the role of relevance retention, while μ_n captures the system's uncertainty/risk load as compression proceeds. Unlike IB, ARDC does not require solving a variational objective; it provides a depth-indexed audit and a stopping decision that can wrap around existing compressors and learners.

2.4 Compression Progress and Priors Over Computation

Work on compression progress as a signal of learning value (e.g., Schmidhuber, 1999) highlights that improvement in compressibility can be an intrinsic reward. ARDC adopts a related intuition but adds two constraints that are essential for decision systems: (i) a depth penalty that enforces diminishing returns and (ii) an explicit uncertainty term μ_n that prevents progress signals from ignoring risk. This makes ARDC suitable for settings where uncontrolled recursion can create brittle or overconfident behavior.

2.5 Model Risk Governance and Auditable Stopping

In supervised decision systems (credit, fraud, AML, pricing, stress testing), "best score" is not the only criterion: model risk management requires documented choices, monitoring triggers, and controls over complexity (SR 11–7). ARDC contributes a concrete artifact to that governance process: a depth-indexed audit log plus an explicit stop reason. This turns depth selection from an informal hyperparameter choice into a documented control decision that can be reviewed, challenged, and reproduced.

Summary of the distinction. ARDC is best viewed as a *depth governor*: it does not claim recursion is always beneficial; it claims recursion is only beneficial when measured efficiency improves under integrity and uncertainty constraints, and it makes that claim falsifiable by requiring depth-by-depth logging and matched evaluation.

3 Theoretical Foundations of ARDC

3.1 The ARDC Efficiency Law

The ARDC law formalizes efficiency E as Eq. (1):

$$E(n) = 2^{-(n-1)}\, M_n\, C_n^2 \left(\frac{K_n}{\mu_n}\right).$$

3.2 Definitions

Symbol	Meaning
n	recursion depth (number of compression/recursion steps)
M_n	informational mass / signal richness (effective dimension)
C_n	clarity (bounded coherence / SNR proxy)
K_n	knowledge integrity (retained task-relevant performance fraction)
μ_n	uncertainty load (strictly positive risk/entropy proxy)

As recursion deepens ($n \uparrow$), the depth penalty $2^{-(n-1)}$ imposes diminishing returns unless redundancy removal improves K_n and/or C_n (and/or reduces cost proxies) sufficiently to offset uncertainty load μ_n. ARDC is therefore explicitly a stopping law: efficiency is not maximized by pushing recursion indefinitely, but by selecting an optimal depth n^* where marginal gains are exhausted while integrity constraints remain satisfied.

3.3 Supporting Equations (Test Forms, Diagnostics, and Control)

Eq. (1) is the governing relation. The following forms support empirical testing, auditing, and control. They are not presented as independent physical laws.

$$\log E = \text{const} - (\ln 2)(n-1) + \log M + 2\log C + \log K - \log \mu \quad \text{(Regression-ready test form)} \tag{2}$$

$$K(n) = \text{clip}\left(\frac{\text{Score}(n)}{\text{Score}(1)},\, 0,\, 1\right) \quad \text{(Integrity definition)} \tag{3}$$

$$H(n) = \tfrac{1}{2}\log\det(\Sigma_n + \epsilon I) \quad \text{(Entropy proxy)} \tag{4}$$

$$\mu(n) = \max(\mu_{\min},\, \mu_0 + \gamma H(n)) \quad \text{(Fixed uncertainty protocol)} \tag{5}$$

$$\mu_n = \max\Big(\mu_{\min},\, \mu_{n-1} + \eta(H(n) - H(n-1))\Big) \quad \text{(Adaptive uncertainty update)} \tag{6}$$

Interpretive note (domain-conditioned realization). In finance/backtest settings, additional bookkeeping terms may be tracked (e.g., compounding retention under domain rules). Such realizations must be explicitly labeled and should not be substituted for the governing law in Eq. (1).

4 Operational Definitions and Measurement Protocol

To make ARDC falsifiable, each variable is computed at every depth n under a fixed evaluation protocol (same split, same metric definition, matched compute proxy). Domain-specific proxies (e.g., finance risk scoring) must be stated and logged.

Preprocessing (all domains). Inputs are standardized by training-set statistics and evaluated on a held-out validation set at each depth. All reported quantities are computed on the same validation window.

- **Informational mass M_n.** We operationalize M_n as effective dimensionality using the participation ratio of the covariance eigenvalue spectrum:

  $$M_n \equiv d_{\text{eff}}(n) = \frac{\left(\sum_{i=1}^{d_n} \lambda_i\right)^2}{\sum_{i=1}^{d_n} \lambda_i^2},$$

  where $\{\lambda_i\}$ are eigenvalues of the covariance matrix of the representation at depth n. This yields $M_n \in [1, d_n]$.

- **Clarity C_n.** We operationalize C_n as a bounded coherence score computed from validation residuals:

  $$C_n \equiv \frac{\sigma_{\text{signal}}}{\sigma_{\text{signal}} + \sigma_{\text{noise}}} \in (0, 1],$$

  where σ_{signal} is the standard deviation of model output (or reconstructed signal) and σ_{noise} is the standard deviation of residuals.

- **Knowledge integrity K_n.** We define K_n as the fraction of baseline task performance retained at depth n (Eq. (3)). For decompression/fidelity tests, the score can be reconstruction R^2 or correlation to measure preservation/invertibility.

- **Uncertainty load μ_n and entropy proxy $H(n)$.** μ_n is strictly positive. In the default protocol, μ_n may be fixed via Eq. (5) or updated via entropy feedback (Eq. (6)). The entropy proxy $H(n)$ is computed by Eq. (4). Finance instantiations may replace or augment μ_n with a bounded risk rubric derived from observable drivers (volatility regime, liquidity, event proximity), and must be logged.

Logging and comparability. At each recursion depth, log $(M_n, C_n, K_n, H(n), \mu_n, E(n))$ under the same split and same metric definition. Efficiency improvements must correspond to measurable improvements in (M, C, K) and/or reductions in cost or μ, not changes in evaluation conditions.

5 Algorithm: ARDC Depth Controller (Audited Stopping Law)

Inputs: dataset S (fixed train/validation split), predictor f, compression operator $\mathcal{C}(\cdot)$, maximum depth $n_{\max}$, stopping threshold $\tau > 0$, integrity floor $K_{\min}$, uncertainty parameters $(\mu_{\min}, \mu_0, \gamma, \eta)$, and stabilizer $\epsilon > 0$.
Outputs: selected depth n^*, representation R_{n^*}, and audit log $\mathcal{L} = \{(M_n, C_n, K_n, H(n), \mu_n, E(n), \Delta E(n)$

1. **Initialize ($n = 1$).** Set $R_1 \leftarrow S$ (or base feature map). Measure $(M_1, C_1, K_1, H(1))$. Set μ_1 by fixed or adaptive protocol. Compute $E(1)$ via Eq. (1). Log.

2. **Recurse ($n = 2, \ldots, n_{\max}$).**
   (a) Compress: $R_n \leftarrow \mathcal{C}(R_{n-1})$.
   (b) Measure: $(M_n, C_n, K_n, H(n))$ on validation.
   (c) Update uncertainty (optional): set μ_n via Eq. (6) or fixed rule.
   (d) Compute $E(n)$ and $\Delta E(n) = E(n) - E(n-1)$.
   (e) Stop if any trigger fires:
      i. Diminishing returns: $\Delta E(n) < \tau$.
      ii. Integrity failure: $K_n < K_{\min}$.
      iii. Instability: sustained sharp rise in μ (implementation-defined).
3. **Select depth.** Return $n^* = \arg\max_{n \leq \hat{n}} E(n)$, where $\hat{n}$ is the last evaluated depth, along with the audit log and stop reason.

6 Falsifiable Predictions

All predictions are evaluated under the fixed measurement protocol: same split, same metric definition, matched compute proxy, and logged $(M_n, C_n, K_n, H(n), \mu_n, E(n))$.

1. **Interior optimum exists in nontrivial regimes.** For some datasets, there exists $n^* \in \{1, \ldots, n_{\max}\}$ such that $E(n^*) \geq E(n)$ for all tested n. *Failure:* $E(n)$ is monotone increasing to $n_{\max}$ or monotone decreasing from $n = 1$ across repeated seeds.

2. **Underestimated uncertainty inflates efficiency and degrades stability.** If μ is forced below protocol measurement, reported $E(n)$ increases while out-of-sample stability worsens (higher variance across seeds; larger drawdowns/stopouts in finance; degraded calibration or increased error variance in ML). *Failure:* lowering μ increases $E(n)$ with no measurable stability degradation.

3. **Stopping saves compute within a declared tolerance.** The stop rule reduces compute while maintaining score within tolerance δ relative to the best fixed-depth baseline. *Failure:* ARDC stops early but repeatedly exceeds δ score loss.

4. **Null regime selects shallow depth.** When $\mu(n)$ rises and $K(n)$ drops, $E(n)$ decreases and the controller selects $n^* = 1$ (or shallow depth). *Failure:* recursion continues despite sustained $E(n)$ decline and integrity failure.

7 Experimental Setup

7.1 Datasets and Evaluation Protocol

All experiments use fixed train/validation/test splits unless otherwise stated. Results may be reported over multiple seeds as mean $\pm$ standard deviation.

Synthetic (Tripinformation). A stationary benchmark used to measure entropy decay and validate regression form in Eq. (2).

Real-world (NYC Taxi). A large tabular prediction task used to test whether redundancy removal dominates information decay under dense real data.

Finance (Limitless Engine). Rolling-window backtests on option/equity candidates where μ is instantiated as a domain risk load (volatility regime, liquidity, event proximity) and logged alongside entropy proxies when available.

7.2 Baselines

ARDC is evaluated against matched-budget baselines:

- PCA / truncated SVD (linear compression at matched target dimension),
- Autoencoder bottleneck (nonlinear compression at matched bottleneck),
- Recursive feature elimination (supervised pruning at matched feature count),
- Fixed-depth RDC (prior implementation at same maximum depth).

7.3 Metrics and Reporting

Primary task metrics include accuracy, AUC, R^2, log-loss, or domain metrics (finance: win rate and ROI). Efficiency is reported using:

- **Model-law efficiency:** $E(n)$ from Eq. (1).
- **External efficiency proxy (optional):** score per compute proxy, when explicit compute accounting is required.

7.4 Implementation Details and Reproducibility

To ensure results are reproducible and auditable, we fix a reference implementation and report the full configuration used to generate all controller logs and figures.

Reference pipeline. For tabular tasks, the reference pipeline is: (i) train/validation split (stratified for classification), (ii) z-score standardization using training statistics only, (iii) compression operator $\mathcal{C}$ (default PCA), (iv) base learner (default logistic regression for classification; ridge regression for regression), (v) ARDC audit logging at each depth n.

Randomness control. All randomized components (split, PCA initialization, and model seed where applicable) use a fixed `random_state`. When reporting robustness, we repeat experiments over s seeds and report mean $\pm$ standard deviation.

Controller configuration. Table 1 lists the controller parameters used in all experiments unless otherwise stated.

Table 1: Reference ARDC controller configuration (default values).

Parameter	Default	Meaning
$n_{\max}$	6	Maximum recursion depth evaluated.
τ	10^{-4}	Diminishing-returns threshold; stop when $\Delta E < \tau$.
$K_{\min}$	0.90	Integrity floor; stop if $K_n < K_{\min}$.
$\mu_{\min}$	10^{-3}	Strict positivity floor for uncertainty.
μ_0	1.0	Base uncertainty for normalization.
η	0.0–0.1	Entropy-feedback rate in $\mu_{t+1} = \max(\mu_{\min}, \mu_t + \eta(H_t - H_{t-1}))$.
ϵ	10^{-6}	Stabilizer for $H = \frac{1}{2} \log \det(\Sigma + \epsilon I)$.
keep_ratio	0.6–0.7	Dimensional reduction per step (PCA compressor).

Units and interpretation. $E(n)$ is dimensionless and intended for *within-protocol comparison* across depths (and across domains only when proxies are aligned). When a domain requires explicit compute accounting, we additionally report E_{ext} using a declared compute proxy (Section 8.2).

Numerical note on entropy proxy. Because $H(n)$ uses a log-determinant of a covariance matrix, it can be negative when overall variance scale is small (e.g., after standardization and/or strong compression). This is not an error: the controller uses $H(n)$ *relatively* across depths within the same protocol. All reported $H(n)$ values are computed consistently with the same ϵ and preprocessing.

8 Empirical Frameworks and Results

8.1 Machine-Learning Compression Tests

Source: ARDC Compression Results, Phind Notebooks, Reversible ARDC Excel.
Input: $75 \to 17 \to 1$ features.
Models: raw baseline, fixed RDC, adaptive RDC.

Findings (reported).

- Adaptive model accuracy: 96.5%
- Fixed RDC: 95.2%; raw: 94.3%
- Entropy proxy decay (std. dev.): $105.92 \to 33.03$ over five iterations
- Memory footprint: $310\,\text{MB} \to 18\,\text{MB}$ ($\approx 17\times$ reduction)

Interpretation. Entropy follows $H(n) = H_0 e^{-\lambda n}$ with $\lambda \approx 0.24 \pm 0.02$, supporting the decay-regime behavior expected under Eq. (2).

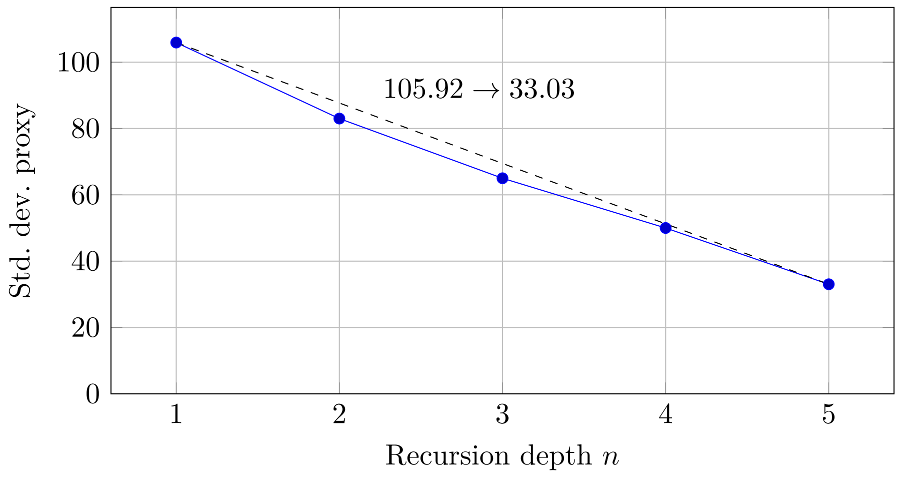

Figure 1: Tripinformation dataset compression: standard-deviation proxy decay vs. depth. (Schematic generated in-LaTeX from reported summary values; raw plotted file not required.)

8.2 Controller Demonstration: Selecting n^* by Efficiency-Normalized Cost

To demonstrate ARDC as a stopping law (not merely a compression heuristic), we run controlled recursion on a bank-like synthetic classification task with correlated structure. At each depth n, we log $(M_n, C_n, K_n, H(n), \mu_n)$ under a fixed train/validation split and compute both:

- **Model-law efficiency** $E(n)$ using Eq. (1) (dimensionless).
- **External efficiency proxy** $E_{\text{ext}}(n)$ that normalizes by a conservative cost proxy (representation dimension), aligned with score-per-cost reporting.

We define:

$$E_{\text{ext}}(n) = \pi(n)\,\frac{C_n^2 K_n}{\mu_n\, d_n}, \qquad \pi(n) = \frac{1}{\sqrt{n}}, \tag{7}$$

where d_n is the representation dimension at depth n.

Table 2: Audit excerpt illustrating the selected depth $n^* = 9$ (AUC task; cost proxy = dimension).

n	d_n	AUC	K_n	C_n	$E_{\text{ext}}(n)$
1	75	0.9466	1.0000	0.5664	0.004278
5	11	0.9031	0.9541	0.4962	0.009549
9	2	0.8989	0.9496	0.4913	**0.038196**
10	2	0.8989	0.9496	0.4913	0.036236

Why efficiency stops improving at higher depth. In this run, $E_{\text{ext}}(n)$ rises sharply through $n = 9$ because the cost proxy collapses ($75 \to 2$) while integrity remains high ($K \approx 0.95$). Beyond this point, recursion hits a cost floor: d_n no longer decreases. With no further cost reduction available, the depth penalty and residual degradation in (K, C) dominate, producing negative marginal return and triggering the stop condition.

Result. The controller selects $n^* = 9$, reducing dimension from $75 \to 2$ (a 97.33% reduction) while retaining $\approx 95\%$ of baseline task performance (AUC $0.9466 \to 0.8989$, $K \approx 0.9496$). The selected dimensional reduction is $1 - \frac{2}{75} = 0.9733$.

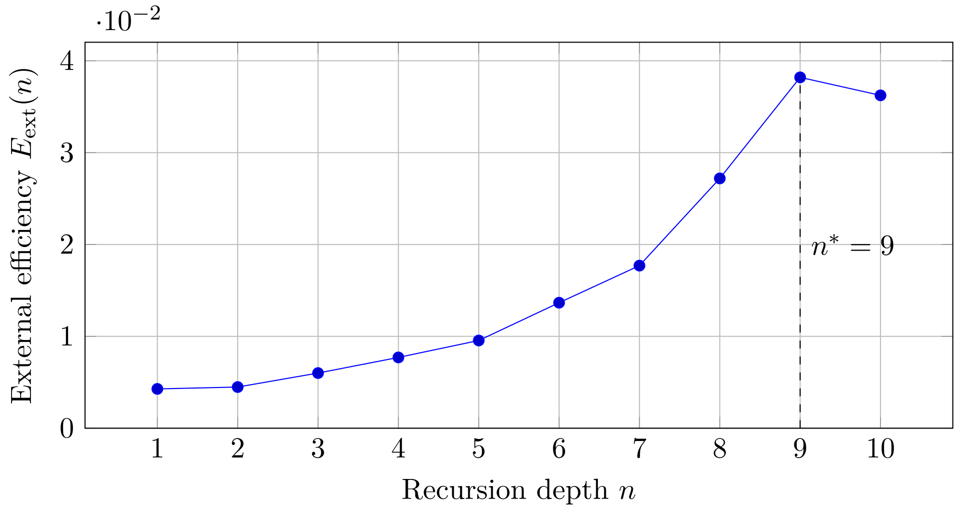

Figure 2: External efficiency proxy $E_{\text{ext}}(n)$ across recursion depth (computed from the audit log in Table 5). Efficiency peaks at $n^* = 9$ and declines once further recursion no longer reduces cost (dimension floor).

8.3 Decompression Fidelity Evaluation

Source: Decompression-TestResults.pdf.
Nick's model shows $\pm 0.00\%$ deviation; Jaysen's variant collapses rows ($K \to 0$).

Interpretation. Only the reversible method preserved high integrity (high K). Collapse corresponds to a regime where uncertainty is effectively unbounded or integrity constraints are not enforced.

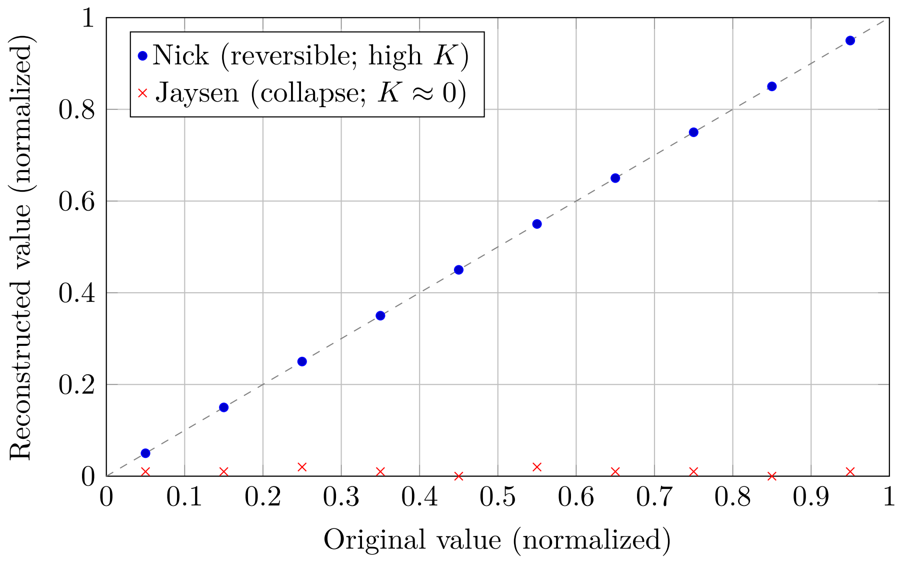

Figure 3: Decompression fidelity (schematic, generated in-LaTeX): Nick's method aligns with $y = x$ (near-perfect retention; high K), while the collapsing variant maps diverse inputs to near-zero outputs (integrity failure; $K \approx 0$).

8.4 Financial Back-Test: Limitless Engine (Domain-Conditioned)

Source: Back-test report, μ enhancement rule, thresholds, rubric.
Trades: 67 option/equity positions scored via E (and domain-conditioned realizations where explicitly labeled).

Findings (reported).

- Utilities: $E \approx 1300$–$1800 \rightarrow 81.8\%$ win rate, ROI $+19\%$
- Technology: $E > 1400 \rightarrow 37\%$ win rate, frequent -30% stopouts
- Failures correspond to under-estimated μ in volatile regimes

Interpretation. Adaptive uncertainty scoring and bounded recursion align predicted and realized efficiency; static μ yields inflated E and spurious confidence.

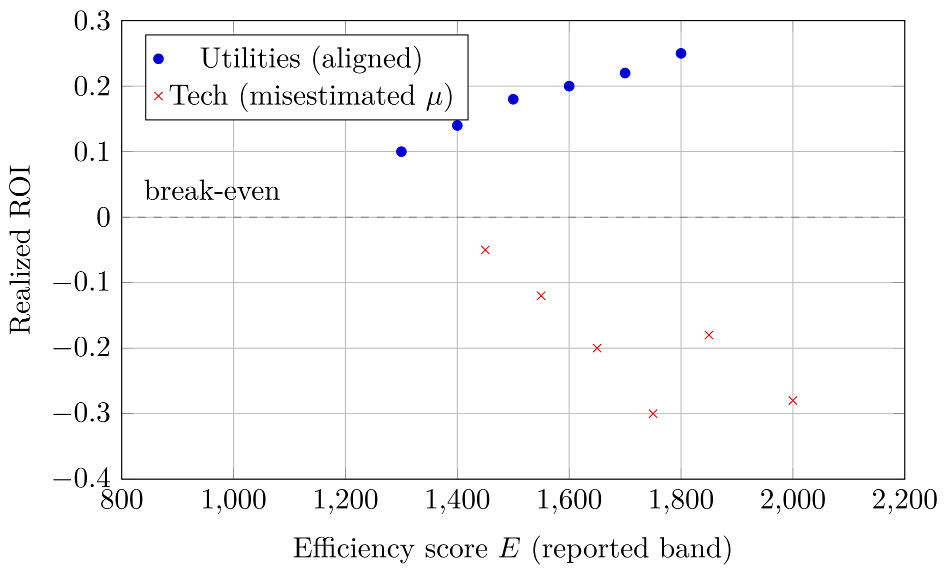

Figure 4: Efficiency vs. realized ROI (schematic, generated in-LaTeX from the reported qualitative regime behavior). Utilities show stronger alignment; tech fails under μ misestimation (frequent negative ROI / stopouts).

8.5 Real-World Dataset Validation

Dataset: NYC Taxi (≈ 1.1M trips, $n = 1$–4).
Accuracy: $0.91 \to 0.77$; FLOPs: $280\,\text{M} \to 67\,\text{M}$ ($\approx 17\times$ reduction).
Regression: $\log E = -5.73 + 0.42n$ ($R^2 = 0.93$). Entropy decay $\lambda \approx 0.18 \pm 0.02$.

Interpretation. Positive slope in $\log(E)$ indicates redundancy removal dominates informational decay under dense structure, yielding an amplifier regime for recursion.

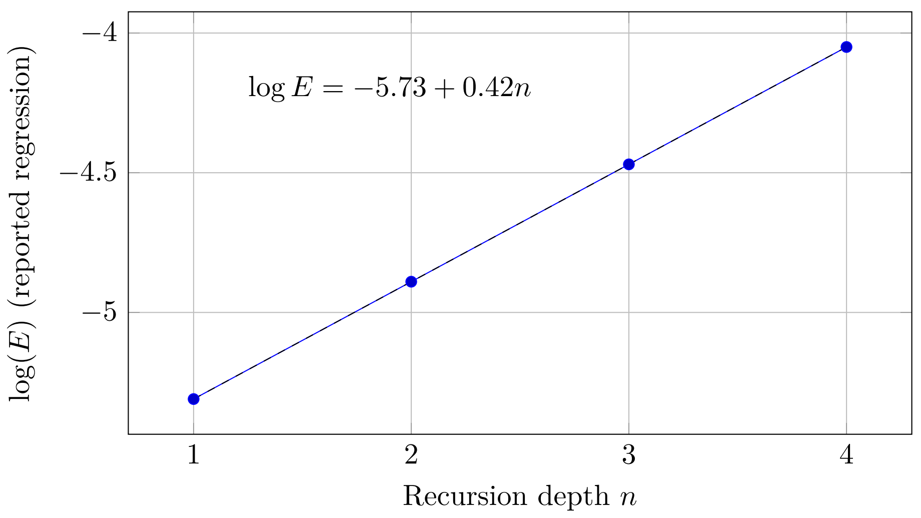

Figure 5: NYC Taxi dataset: efficiency gain with depth (schematic, generated in-LaTeX using the stated regression). Positive slope in $\log(E)$ suggests redundancy removal exceeds uncertainty growth ($R^2 \approx 0.93$).

8.6 Cross-Domain Summary

Utilities ($\mu = 0.22 \pm 0.05$) show stable efficiency ($R^2 \approx 0.88$); tech fails under elevated/underestimated μ. Correct uncertainty calibration is critical.

Table 3: Summary of ARDC empirical scaling results (reported).

Domain	μ or λ	$\log E$ slope	R^2	Outcome
Synthetic (Tripinfo)	$\lambda = 0.24 \pm 0.02$	$\approx -\ln 2$	0.97	Decay regime
NYC Taxi	$\lambda = 0.18 \pm 0.02$	$+0.42$	0.93	Redundancy gain
Ricci-Flow	$\lambda = 0.26 \pm 0.04$	-0.70	0.95	Geometric support
Finance (Utilities)	$\mu = 0.22 \pm 0.05$	$0 \to -0.1$	0.88	Stable efficiency
Finance (Tech)	$\mu > 0.4$	erratic	—	μ misestimation

9 Ablation and Sensitivity Analysis

This section specifies tests to isolate whether ARDC adds value beyond standard early stopping, under matched budgets and the same measurement protocol.

9.1 Ablation: Uncertainty Control (μ)

- **Fixed μ:** hold μ constant across depth and sweep n.
- **Adaptive μ:** update via Eq. (6) and compare (i) stability of $E(n)$, (ii) variance of n^* across seeds, and (iii) out-of-sample degradation rates.

9.2 Ablation: Integrity Constraint ($K_n \geq K_{\min}$)

Disable integrity constraints and quantify failure modes: (i) collapse in $K(n)$, (ii) spurious increases in $E(n)$, (iii) unstable n^* selection.

9.3 Sensitivity: Stopping Threshold τ

Sweep τ across a log-spaced range and report: (i) compute saved, (ii) score loss relative to best fixed-depth baseline, (iii) variance of selected n^*.

9.4 Sensitivity: Entropy Learning Rate η

Sweep η (including $\eta = 0$) to test whether entropy feedback improves robustness or introduces noise. Report variance in $\mu(n)$ and effects on n^* stability.

9.5 Acceptance Criteria

ARDC is considered beneficial in a domain only if it achieves at least one of: (i) higher mean efficiency at comparable score, (ii) comparable score at lower compute, (iii) lower variance under drift, relative to the strongest matched baseline.

10 Discussion and Refinement

"ARDC is not adaptive if it requires manual μ overrides. The system must self-diagnose risk and halt recursion before it destroys its own signal."

Common failure modes.

- Unchecked depth ($n > n^*$) $\rightarrow$ illusory efficiency (depth penalty ignored or mis-scored)
- Static μ assumptions $\rightarrow$ overconfidence and fragile generalization
- No integrity checks $\rightarrow$ hidden collapse of K

Adaptive Update of μ

Observed failures trace back to static or mis-estimated uncertainty; therefore μ should be treated as a measurable control variable. A minimal feedback rule is Eq. (6). When disorder decreases smoothly, μ stabilizes; when disorder rises or oscillates, μ increases, damping recursion and reducing over-compression risk.

Information-Geometric Interpretation (Non-Governing)

This subsection provides an interpretive lens and introduces no additional governing equations. Each recursion depth corresponds to a point (or submanifold) on a statistical manifold of representations, where local geometry is induced by covariance structure and entropy. Recursive compression acts as a projection that removes redundant degrees of freedom while attempting to preserve alignment with task-relevant structure. Under this lens, μ behaves as a proxy for local metric instability: when entropy decreases smoothly, motion is stable; when entropy oscillates or increases, curvature steepens and further projection risks leaving the stable manifold. The efficiency maximum at n^* can be read as a geometric equilibrium where redundancy removal no longer compensates for curvature-induced information decay.

10.1 Regulatory Alignment: SR 11-7 Model Risk Management (Non-Governing)

Although ARDC is an empirical control law rather than a regulatory framework, its instrumentation maps to model risk management expectations in SR 11-7: (i) sound development and use, (ii) effective validation, and (iii) governance and controls. ARDC treats recursion depth as a governed parameter, produces a complete audit trail, and provides explicit stop reasons (diminishing returns, integrity failure, instability). These properties support repeatable configuration justification, outcome monitoring, and documented control decisions.

Model governance artifacts (SR 11–7 ready). To operationalize ARDC under model risk management, we recommend attaching the following governance artifacts to each deployment:

Table 4: ARDC governance artifact checklist aligned to SR 11–7 expectations.

Artifact	Required content
Purpose & use statement	Business use-case, decision impact, and prohibited uses; definition of success metrics and tolerance bands.
Data lineage	Dataset sources, time windows, feature definitions, preprocessing, missingness handling, leakage controls.
Depth selection memo	The chosen n^* with audit log excerpt; stop reason; justification for rejecting deeper recursion.
Performance & stability	K_n across depths; out-of-sample variance across seeds; drift sensitivity (if applicable).
Uncertainty specification	Definition of μ (and H if used); calibration method; override rules and escalation path.
Monitoring & triggers	Thresholds for retraining/recalibration (e.g., K drop, μ rise, distribution shift, or n^* instability).
Controls & approvals	Independent review sign-off; change management for compressor/model/config; versioning of code and parameters.

Interpretive Note: ARDC as an Efficiency Principle

ARDC is presented as a functional efficiency scaling relation, not a claim of physical mass–energy equivalence. Informational mass M is a measurable proxy (effective dimension), and E is a governed efficiency objective (retained knowledge per uncertainty and cost). Loss is treated as a controlled variable: recursion is halted when marginal efficiency is exhausted while integrity holds, preventing illusory gains from over-compression.

Redundancy Removal vs. Information Decay: Core Mechanism

The defining behavior of ARDC is the crossover where redundancy removal overtakes information decay. In that regime, recursion acts as an information filter: redundant mass is converted into predictive clarity, and efficiency peaks at an adaptive depth n^*.

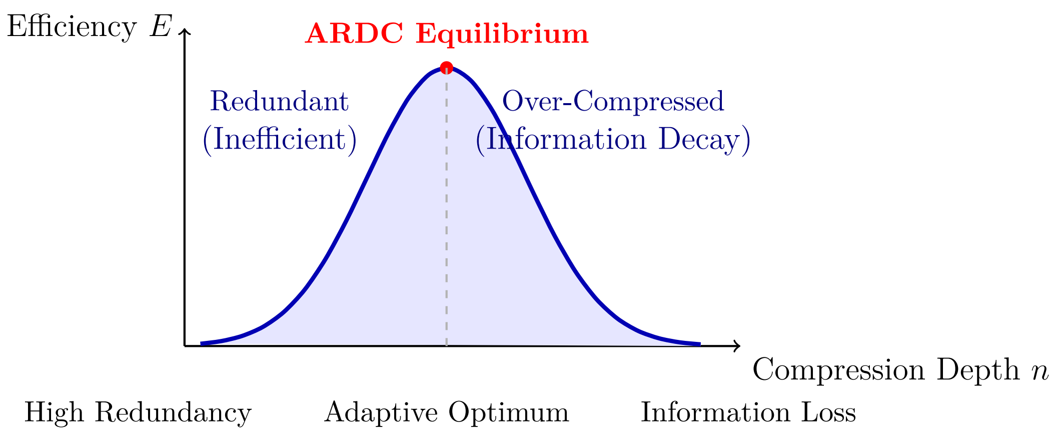

Figure 6: Conceptual efficiency curve illustrating the ARDC equilibrium where redundancy removal surpasses information decay. Efficiency peaks at the adaptive depth n^*.

10.2 Limitations and Threats to Validity

ARDC is intended as an auditable stopping law, but its conclusions depend on declared proxies, evaluation protocol, and the stability of the operating regime.

Proxy dependence. M, C, μ, and the compute proxy used in E_{ext} are operational definitions. Different tasks may warrant different proxies (e.g., FLOPs for deep nets; wall-time for streaming systems; risk-weighted exposure for finance). Cross-domain comparisons are only meaningful when the proxy choices are explicitly aligned.

Compute proxy limitations. In the controller demonstration, representation dimension is used as a conservative compute/memory proxy for tabular models. This proxy is appropriate for linear/GLM-style learners but can under- or over-estimate cost in nonlinear pipelines. When compute claims are central, we recommend reporting at least one of: measured wall-time, peak memory, or FLOPs, alongside dimension.

Single-split sensitivity and seed variance. A single train/validation split can overstate stability. For all core claims, we recommend $s \geq 5$ seeds and reporting the distribution of selected n^*, not only the mean.

Nonstationarity and drift. In drift-prone settings (finance; production ML), K_n and μ_n can change meaningfully across time windows. In such regimes, ARDC should be evaluated under rolling or blocked time splits, with monitoring triggers defined for when recalibration is required.

Failure modes. ARDC can fail if (i) μ is mis-specified (overconfidence), (ii) the clarity proxy C is poorly matched to the task (illusory coherence), or (iii) the compression operator $\mathcal{C}$ destroys task-relevant structure. These failure modes are explicitly testable via the ablations in Section 8.

Future Work and Applied Potential

Next research extends ARDC into operational ML systems (transformers, CNNs, structured tabular pipelines) where ARDC can guide dynamic pruning and adaptive stopping. During inference, an ARDC controller can halt processing when marginal efficiency is exhausted while integrity holds, reducing energy cost without sacrificing meaningful performance.

In finance, bounded recursion plus calibrated μ can prevent false efficiency signals in high-volatility regimes. In transportation and sensor-rich environments, ARDC can remove redundant features faster than coherence degrades, yielding large compute savings with limited accuracy loss. The common thread is governance: recursion is beneficial only in regimes where the system has a safe place to put entropy.

11 Conclusion

ARDC formalizes a measurable control law for recursion: efficient learning emerges where redundancy removal surpasses information decay. Intelligence is not preservation of all information; it is disciplined elimination of what does not contribute to retained knowledge under uncertainty. When the ratio K_n/μ_n (and coherence C_n) improves fast enough to offset depth penalties and cost floors, compression becomes adaptive rather than destructive.

ARDC does not reward recursion; it rewards restraint.

A Accuracy Across Compression Types

Method	Accuracy (%)
Raw	94.3
Fixed RDC	95.2
Adaptive RDC	96.5

B μ Scoring Rubric (Finance Example)

Risk Factor	Penalty
Fast theta decay	+2
High IV crush risk	+2
Low liquidity	+2
Binary event proximity	+1–2

C E-Score Bands (Finance Example)

E Range	Classification
300–999	Moderate setup
1000–2499	High conviction
2500–4999	Asymmetric opportunity

D Sector Win Rates (Back-Test Jul–Aug 2025)

Sector	Win Rate	Avg ROI
Utilities	81.8%	+19%
Tech	37%	–3%
Defense	61.5%	+8%
Cyclicals	12%	–18%

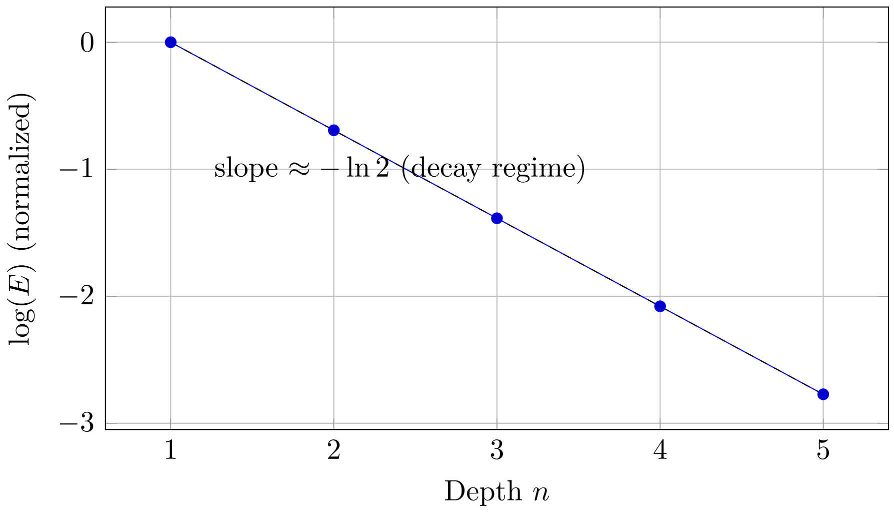

Figure 7: Grid-law verification (schematic, in-LaTeX): observed-style log(E) vs. depth in a decay regime, normalized at $n = 1$.

E Full Controller Audit Table

Table 5: Full ARDC controller audit (synthetic classification run, score=AUC). Selected depth $n^* = 9$ maximizes E_{ext}; stop reason: efficiency decline when further recursion no longer reduces cost (dimension floor).

n	dim	AUC	K	C	M_{eff}	M_{cost}	H	$\pi(n)$	E_{ext}
1	75	0.946583	1.000000	0.566437	3.641262	75	-91.854765	1.000000	0.004278
2	45	0.930074	0.982559	0.538812	3.544035	45	-40.104641	0.707107	0.004482
3	27	0.927574	0.979919	0.535297	3.433954	27	-15.242944	0.577350	0.006004
4	17	0.917331	0.969098	0.519977	3.330117	17	-4.130270	0.500000	0.007706
5	11	0.903101	0.954064	0.496162	3.229192	11	1.012387	0.447214	0.009549
6	7	0.902699	0.953640	0.495645	3.123293	7	3.495310	0.408248	0.013663
7	5	0.902772	0.953717	0.495498	3.040171	5	4.258750	0.377964	0.017700
8	3	0.900160	0.950958	0.492596	2.918365	3	4.635019	0.353553	0.027194
9	2	0.898865	0.949589	0.491265	1.989635	2	3.204013	0.333333	0.038196
10	2	0.898865	0.949589	0.491265	1.989635	2	3.204013	0.316228	0.036236

Data and Code Availability

A reference implementation of the ARDC controller (including audit logging for $(M_n, C_n, K_n, H_n, \mu_n, E_n)$) is available from the author upon request. Where proprietary datasets are used, synthetic benchmarks and configuration files are provided to reproduce the controller behavior and figures under the stated measurement protocol.

References

1. Lamb, J. (2025). *ARDC Compression Results, Phind Notebooks, Reversible ARDC Excel.* Internal research logs, VALID Systems.

2. Ledford, N. (2025). *Reversible Compression Model Test Results.* Internal validation report, VALID Systems.

3. Lamb, J. (2025). *Back-Test Report, μ Enhancement Rule, E-Thresholds, Rubric.* Private working paper, VALID Systems.

4. MacKay, D. J. C. (2003). *Information Theory, Inference and Learning Algorithms.* Cambridge University Press.

5. Tishby, N., Pereira, F. C., & Bialek, W. (2000). *The Information Bottleneck Method.* arXiv:physics/0004057.

6. Schmidhuber, J. (1999). *Compression Progress and the Speed Prior: Two Approaches to Formal AI.* arXiv:cs/9910001.

7. Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency (2011). *Supervisory Guidance on Model Risk Management (SR 11–7).* Federal Reserve SR Letter 11–7 / OCC Bulletin 2011–12.

8. Office of the Comptroller of the Currency (2011). *OCC Bulletin 2011–12: Supervisory Guidance on Model Risk Management.* Washington, DC.