A Hierarchy of Learning Problems: Computational Efficiency Mappings for Optimization Algorithms

Michael Rey *

Posted Date: 11 September 2025

doi: 10.20944/preprints202509.0955.v1

Keywords: optimization; machine learning; operator theory; streaming algorithms; computational complexity

 doi:10.20944/preprints202509.0955.v1

Article

A Hierarchy of Learning Problems: Computational Efficiency Mappings for Optimization Algorithms

Michael Rey

Octonion Group, Hong Kong; contact@octoniongroup.com

Abstract

We establish a rigorous mathematical hierarchy for optimization problems that serves as both theoretical classification and practitioner decision system. Problems are exhaustively categorized into three classes: (1) Non-learnable problems where no alpha-averaged operator exists, precluding iterative convergence; (2) Learnable problems that admit averaged operators enabling batch methods; (3) Streamable problems that are learnable with uniform low-rank residual approximation, enabling rank-K streaming algorithms with explicit bias floors. This framework unifies gradient descent, proximal methods, Frank-Wolfe, PCA, and attention mechanisms under operator-theoretic foundations. We provide explicit computational mappings showing memory reductions from quadratic to linear scaling, rigorous Koopman operator connections, and an algorithmic ULRA-test for streamability verification. For typical problems with dimension one million and streamable rank one hundred, streaming achieves ten-thousand-fold memory reduction with identical convergence rates.

Keywords: optimization; machine learning; operator theory; streaming algorithms; computational complexity

---

1. Introduction

The landscape of modern optimization is characterized by a vast and often bewildering array of algorithms, from classic gradient-based methods to sophisticated streaming and distributed systems. For practitioners and researchers alike, a fundamental challenge persists: how does one select the *right* algorithm for a given problem? More critically, how can we know, *a priori*, the most efficient computational paradigm a problem will admit? Answering this question is the key to unlocking immense performance gains, conserving computational resources, and ultimately determining whether a large-scale problem is feasible to solve at all.

Currently, this selection process often resembles a "black box" trial-and-error, guided by heuristics and empirical performance on related tasks. This work dismantles that black box. We introduce a rigorous, three-tier mathematical hierarchy that classifies any optimization problem based on its intrinsic structural properties. This classification is not merely a theoretical exercise; it forms the foundation of a practical decision framework that allows any user to:

1. **Assess Learnability**: Determine if a problem is fundamentally solvable by iterative methods. A "non-learnable" problem, as we define it, is guaranteed to fail with standard first-order techniques, saving countless hours of fruitless experimentation. This assessment directs efforts towards necessary reformulation or regularization.
2. **Assess Streamability**: For learnable problems, determine if they possess the "streamable" property—a specific low-rank structure in their residuals. This is the gateway to extreme computational efficiency.
3. **Unlock Efficiency Gains**: A positive streamability test guarantees that the problem can be solved with algorithms whose memory and computational footprints are orders of magnitude smaller than their traditional "batch" counterparts. For a problem in dimension $d = 10^6$ with a streamable

 doi:10.20944/preprints202509.0955.v1

rank of $K = 100$, this translates to a staggering $10,000\times$ reduction in memory and per-iteration cost.

This framework provides a clear, structured, and theoretically-grounded roadmap for navigating the complex world of optimization. By moving from heuristic guesswork to a formal assessment of problem structure, we empower practitioners to make optimal algorithmic choices, predict performance, and push the boundaries of what is computationally achievable. This work establishes that hierarchy, proves its foundations in operator theory, and demonstrates its profound practical consequences with concrete examples and complexity mappings..

2. A Practical Decision Framework for Optimization

The theoretical hierarchy we develop is not merely for classification; its primary purpose is to empower practitioners with a clear, actionable decision-making framework. This framework transforms the abstract question of algorithmic choice into a concrete, step-by-step diagnostic process. Before delving into the mathematical underpinnings, we present this practical workflow.

1. **The Learnability Test: Is the Problem Solvable?** The first and most crucial question is whether the problem is *learnable*. In our framework, this means verifying the existence of an associated α-averaged operator with a fixed point. A problem that fails this test is classified as **non-learnable**.
   - **Implication**: A non-learnable problem is guaranteed to diverge or fail to converge with standard iterative first-order methods (like gradient descent and its variants). This is not a statement about a specific algorithm's failure, but a fundamental property of the problem itself.
   - **Action**: Instead of wasting resources on fruitless tuning of a doomed approach, the practitioner knows immediately that the problem must be fundamentally altered. The only paths forward are through *regularization* (to smooth the problem), *reformulation* (to create a different but related problem), or *feature lifting* (to embed the problem in a higher-dimensional space where it becomes learnable).
2. **The Streamability Test: Is Extreme Efficiency Possible?** If a problem passes the learnability test, it is guaranteed to be solvable with a traditional *batch* algorithm (e.g., using the full dataset or a dense matrix operator at each step). The next question is whether we can do dramatically better. This is the streamability test. A problem is **streamable** if its underlying residual operator admits a uniform low-rank approximation. We provide a concrete numerical procedure, the ULRA-Test (Algorithm 10), to verify this property.
   - **Implication**: A positive streamability test is a gateway to enormous computational savings. It certifies that the problem can be solved using a streaming algorithm that only requires a small, rank-K portion of the operator at each step.
   - **Action**: If the problem is streamable, the practitioner can confidently select a streaming algorithm, knowing it will converge and knowing the trade-offs.
3. **Algorithm Selection: Matching the Algorithm to the Problem DNA** The results of these two tests dictate the optimal algorithmic path:
   - If **Non-Learnable**: The only choice is to go back and modify the problem itself. No off-the-shelf optimization algorithm will work.
   - If **Learnable but Not Streamable**: The problem is solvable, but likely requires batch methods. The practitioner should budget for memory and computation proportional to $O(d^2)$ for a d-dimensional problem. This is a perfectly valid and often necessary approach for many problems.
   - If **Streamable at Rank** K: The practitioner now has a choice, unlocking the possibility of extreme efficiency gains. The decision can be based on specific constraints:

 doi:10.20944/preprints202509.0955.v1

- **Memory-Constrained Environments**: Choose the streaming algorithm. Its memory footprint will be $O(Kd)$ instead of $O(d^2)$—a potential reduction from terabytes to megabytes.
- **Communication-Limited Settings (e.g., Federated Learning)**: The low-rank structure allows for massive compression of updates, making distributed training feasible over slow networks.
- **When Maximum Accuracy is Paramount**: If the small approximation error (ε_K) of the streaming method is unacceptable for the final application, the practitioner can still fall back to the batch method, making a fully informed decision about the cost-accuracy trade-off.

This framework replaces heuristic guesswork with a rigorous, predictive, and actionable workflow.

3. Mathematical Preliminaries

Let $(V, \langle\cdot,\cdot\rangle)$ be a finite-dimensional Hilbert space over $\mathbb{R}$, $\mathbb{C}$, or $\mathbb{H}$ (quaternions). Consider an optimization problem with objective $R : V \to \mathbb{R}$ and feasible set $D \subset V$ closed and convex. We denote solutions by $\theta^\star \in \arg\min_{\theta\in D} R(\theta)$.

We do not assume differentiability or smoothness upfront. Instead, we characterize solvability through operator theory.

Definition 1 (Averaged Operator). *An operator $T : D \to D$ is α-averaged for $\alpha \in (0,1)$ if there exists a nonexpansive operator $S : D \to D$ such that $T = (1-\alpha)I + \alpha S$.*

4. The Three-Tier Hierarchy

Definition 2 (Problem Classification). 1. *A problem (R, D) is **non-learnable** if no α-averaged operator $T : D \to D$ exists with a fixed point $\theta^\star \in D$ such that $\theta_{t+1} = T(\theta_t)$ converges.*
2. *A problem (R, D) is **learnable** if there exists an α-averaged operator $T : D \to D$ with fixed point $\theta^\star \in D$ such that the iteration $\theta_{t+1} = T(\theta_t)$ converges to $\theta^\star$ for all $\theta_0 \in D$.*
3. *A problem (R, D) is **streamable at rank** K if it is learnable and the residual mapping $r(\theta) := \theta - T(\theta)$ satisfies*

$$\sup_{\theta\in D} \left\| r(\theta) - \sum_{i=1}^{K} g_i(\theta) \otimes h_i(\theta)^* \right\| \leq \varepsilon_K \tag{1}$$

for bounded maps $g_i, h_i : D \to V$ and some $\varepsilon_K \geq 0$.

Theorem 1 (Hierarchy Characterization). *The following strict inclusions hold:*

$$\textit{Streamable} \subsetneq \textit{Learnable} \subsetneq \textit{All Problems} \tag{2}$$

Moreover:

1. *Every learnable problem admits a convergent batch method*
2. *Every streamable problem admits both batch and streaming solutions*
3. *Non-learnable problems admit no iterative solution under first-order methods*

Proof. The inclusions follow by definition. For strictness:

Learnable $\not\subseteq$ **Streamable**: Dense logistic regression is learnable via proximal gradient but has full-rank gradients, violating uniform low-rank approximation.

All $\not\subseteq$ **Learnable**: The XOR function in $\mathbb{R}^2$ without feature lifting admits no averaged contraction operator.

Convergence guarantees follow from Krasnoselśkiĭ-Mann iteration theory for averaged operators [1]. □

 doi:10.20944/preprints202509.0955.v1

5. Fundamental Theorems

Theorem 2 (Fundamental Learnability Theorem). *A problem (R, D) is learnable if and only if it admits an α-averaged operator $T : D \to D$ with a fixed point. In particular, learnability immediately implies the existence of a convergent batch method.*

Assumptions.

The equivalence is stated for finite-dimensional Hilbert spaces with closed, convex feasible set D. Under standard first-order schemes (e.g., gradient/proximal/primal–dual), convergence implies the existence of an α-averaged decomposition $T = (1-\alpha)I + \alpha S$ with S nonexpansive (cf. [1,2]).

Proof. The forward direction follows by definition. For the converse, suppose a convergent batch method exists. This implies the existence of an operator T (e.g., gradient descent, proximal gradient) that can be written as $T = (1-\alpha)I + \alpha S$ for some nonexpansive S and $\alpha \in (0, 1)$, with convergent fixed-point iterations. □

Remark 1 (Scope of Learnability). *This definition subsumes gradient descent, proximal methods (ISTA/FISTA), projected subgradient, Frank-Wolfe/conditional gradient [3], and primal-dual operator-splitting (PDHG, Chambolle-Pock). Thus, convex non-smooth problems are naturally included.*

Theorem 3 (Streaming Duality Theorem). *If a problem is streamable at rank K with constant ε_K, then there exists a rank-K streaming algorithm*

$$\theta_{t+1} = \theta_t - \eta \sum_{i=1}^{K} g_i(\theta_t, \xi_t)\langle h_i(\theta_t, \xi_t), \theta_t\rangle \tag{3}$$

with unbiased estimators $\mathbb{E}[g_i(\theta, \xi)] = g_i(\theta)$, $\mathbb{E}[h_i(\theta, \xi)] = h_i(\theta)$, achieving

$$\mathbb{E}[R(\theta_T) - R^\star] \le (1 - \mu\eta)^T [R(\theta_0) - R^\star] + \frac{\eta\sigma^2}{\mu} + \frac{C^2\varepsilon_K^2}{\mu} \tag{4}$$

where μ is the contraction modulus, σ^2 bounds estimator variance, and C bounds factor norms. In particular, $\limsup_{T\to\infty} \mathbb{E}[R(\theta_T) - R^\star] \le c_1\,\sigma^2/\mu + c_2\,\varepsilon_K^2/\mu$.

Proof. The streaming update can be written as $\theta_{t+1} = T(\theta_t) + e_t + \xi_t$ where e_t is the approximation error with $\mathbb{E}[\|e_t\|^2] \le \varepsilon_K^2$ and ξ_t is stochastic noise with $\mathbb{E}[\|\xi_t\|^2] \le \sigma^2$.

Applying standard convergence analysis for averaged operators with additive noise:

$$\mathbb{E}[\|\theta_{t+1} - \theta^*\|^2] \le (1-\alpha)\mathbb{E}[\|\theta_t - \theta^*\|^2] + 2(\varepsilon_K^2 + \sigma^2) \tag{5}$$

Converting to function values using the contraction property completes the proof. □

6. Koopman Operator Connection

Lemma 1 (Koopman-Learnability Equivalence). *Consider a dynamical system $x_{t+1} = f(x_t)$ with Koopman operator $\mathcal{K}$ [4] and finite-dimensional EDMD approximation $K \in \mathbb{R}^{d\times d}$ [5]. The following are equivalent:*

1. *The system is learnable in the EDMD coordinates*
2. *K has spectral radius $\rho(K) < 1$ (contractive dynamics)*
3. *The system is streamable at rank r if K has effective rank r*

Proof. (1) ⇔ (2): Learnability requires convergent fixed-point iteration, which occurs iff the linear operator K is contractive, i.e., $\rho(K) < 1$.

(2) ⇒ (3): If K has effective rank r with $\rho(K) < 1$, then $K = \sum_{i=1}^{r} \sigma_i u_i v_i^T + R$ where $\|\sigma_i\| < 1$ and $\|R\|$ is small. The residual $I - K$ admits rank-r approximation with error $\|R\|$.

 doi:10.20944/preprints202509.0955.v1

(3) $\Rightarrow$ (2): Streamability implies bounded residual approximation, which requires $\rho(K) < 1$ for convergence. □

Remark 2 (Koopman Operator Interpretation). *Recent work on Koopman operators shows that nonlinear dynamics can be lifted to infinite-dimensional spaces where they evolve linearly. In our framework, such problems are learnable precisely when the Koopman operator is contractive (α-averaged). Streamable problems correspond to those where the Koopman operator admits uniform finite-rank approximation, as in EDMD. Without contractivity, the Koopman operator may be unitary or expansive, leading to oscillations or divergence—our definition enforces convergence guarantees.*

7. Complete Problem Classification with Applications

Table 1. Complete hierarchy with computational efficiency mappings and applications

Problem Class	Batch	Streaming	Typical Applications
Streamable Subclass			
PCA / Oja's algorithm [6]	✓	✓ (rank-1)	Dimensionality reduction
Matrix completion (Frank-Wolfe) [3]	✓	✓ (rank-1)	Recommender systems
Separable proximal (L1)	✓	✓ (coord.)	Sparse regression
Linearized attention [7]	✓	✓ (approx)	Transformer efficiency
PowerSGD compression [8]	✓	✓ (rank-K)	Distributed training
Federated learning	✓	✓ (compress)	Edge computing
Learnable but Non-Streamable			
Dense logistic regression	✓	×	Classification
PDE discretizations	✓	×	Scientific computing
General convex problems	✓	×	Optimization
Non-Learnable			
XOR in raw features	×	×	Feature engineering needed
$\tau \to 0$ softmax	×	×	Regularization required
Chaotic dynamics	×	×	Stabilization needed
Discontinuous functions	×	×	Smoothing required

8. Canonical Examples with Concrete Benefits

Proposition 1 (PCA via Oja's Algorithm). *The principal component analysis problem* $\min_{\|w\|=1} -w^T\Sigma w$ *is streamable at rank 1 with* $\varepsilon_1 = 0$.

***Concrete Benefits**: For covariance matrix* $\Sigma \in \mathbb{R}^{d\times d}$ *with* $d = 10^6$*:*

- *Memory: Batch requires 8 TB (storing* Σ*), streaming requires 8 MB (two vectors)*
- *Computation: Batch* $O(d^2) = 10^{12}$ *ops, streaming* $O(d) = 10^6$ *ops per iteration*

Proof. The gradient step with learning rate η gives residual

$$r(w) = \eta(\Sigma w - (w^T\Sigma w)w) = \eta\Sigma w\langle w, w\rangle - \eta(w^T\Sigma w)w \tag{6}$$

This admits exact rank-1 representation: $r(w) = g_1(w)\langle h_1(w), w\rangle$ with $g_1(w) = \eta\Sigma w$ and $h_1(w) = w$, giving $\varepsilon_1 = 0$. □

Proposition 2 (Transformer Attention Linearization). *Linearized attention mechanisms [7] are streamable at rank K with* ε_K *depending on feature map quality.*

***Concrete Benefits**: For attention with sequence length* $n = 2048$*, dimension* $d = 1024$*:*

- *Memory: Standard attention* $O(n^2) = 4$ *MB, linearized* $O(nd) = 2$ *MB*
- *Computation:* $O(n^2d) = 4 \times 10^9$ *vs* $O(nd^2) = 2 \times 10^9$ *operations*
- *Scaling: Linear vs quadratic in sequence length*

 doi:10.20944/preprints202509.0955.v1

Proposition 3 (PowerSGD in Distributed Training)**.** *Distributed optimization with PowerSGD compression [8] is streamable at compression rank K.*

***Concrete Benefits**: For neural network with 10^8 parameters, rank $K = 1024$:*

- *Communication: Full gradients 400 MB, compressed 4 MB (100× reduction)*
- *Bandwidth: Enables training over slow networks (10 Mbps vs 1 Gbps required)*
- *Convergence: Maintains linear rates with bias floor $O(\varepsilon_K^2)$*

Dense Logistic Regression (Learnable but Non-Streamable).

Let $R(\theta) = \mathbb{E}[\log(1+\exp(-y\langle x,\theta\rangle))] + \lambda\|\theta\|_2^2$ with dense, isotropic $x \in \mathbb{R}^d$. Batch (prox-)gradient is convergent (learnable). However, on any compact D containing a neighborhood of $\theta^\star$, $\nabla R(\theta)$ has effectively full rank across directions with a flat spectral tail, so for any fixed $K \ll d$ and any uniform rank-K approximant of the residual $r(\theta)$, the approximation error satisfies $\inf_{\text{rank}\le K} \|r(\theta) - \sum_{i=1}^K g_i(\theta)\otimes h_i(\theta)^*\| \ge c > 0$ uniformly on D. Thus $\varepsilon_K \ge c$ and fixed-rank streaming cannot converge to the optimum (non-streamable), while batch does.

9. The Payoff: Extreme Efficiency Gains from Streamability

The practical importance of our classification hierarchy culminates in the dramatic computational savings unlocked by identifying a problem as streamable. The distinction between a merely learnable problem and a streamable one is the difference between an algorithm that is theoretically solvable and one that is practically feasible at massive scale. This section quantifies those gains.

Theorem 4 (Complexity Mapping for Streamable Problems)**.** *For a problem in a d-dimensional space that is identified as streamable at rank $K \ll d$, the computational complexities of batch versus streaming algorithms are as follows:*

Memory Complexity (Storage of the Operator/Model):

- ***Batch Methods**: Require storing a dense operator, typically a $d \times d$ matrix. The memory complexity is $O(d^2)$.*
- ***Streaming Methods**: Only require storing the components of the rank-K factorization. The memory complexity is $O(Kd)$.*

Per-Iteration Computational Complexity:

- ***Batch Updates**: Involve operations with the dense operator, such as matrix-vector products. The computational cost is $O(d^2)$.*
- ***Streaming Updates**: Involve operations only with the low-rank components. The computational cost is $O(Kd)$.*

9.1. The Practical Impact: From Terabytes to Megabytes, From Days to Seconds

The abstract $O(d^2)$ vs. $O(Kd)$ comparison belies the staggering real-world implications. Let's consider a high-dimensional problem, typical in modern machine learning (e.g., natural language processing, genomics, or recommender systems).

Scenario: A problem with dimension $d = 1,000,000$ (10^6), which is common for large models. Let's assume it is streamable at a rank $K = 100$.

- **Memory Savings**:
  - *Batch Requirement*: Storing a $10^6 \times 10^6$ matrix of double-precision floats (8 bytes each) requires $8 \times (10^6)^2 = 8 \times 10^{12}$ bytes, which is **8 Terabytes (TB)**. This is beyond the capacity of all but the most specialized and expensive supercomputing nodes.
  - *Streaming Requirement*: Storing the rank-100 factors requires approximately $2 \times K \times d \times 8$ bytes, which is $2 \times 100 \times 10^6 \times 8 = 1.6 \times 10^9$ bytes, or **1.6 Gigabytes (GB)**. This fits comfortably in the RAM of a standard laptop.

 doi:10.20944/preprints202509.0955.v1

The memory reduction factor is $\frac{d^2}{Kd} = \frac{d}{K} = \frac{10^6}{100} = 10,000\times$. This is the difference between a problem being fundamentally impossible on most hardware and being easily manageable.

- **Computational Speedup**:
  - *Batch Requirement*: A single iteration involving a dense matrix-vector product would require roughly $(10^6)^2 = 10^{12}$ floating-point operations (FLOPs). On a processor capable of 100 GFLOPs (a reasonable estimate for a high-end CPU core), this single step would take $10^{12}/10^{11} = 10$ seconds.
  - *Streaming Requirement*: A single iteration using the low-rank approximation requires roughly $2 \times K \times d = 2 \times 100 \times 10^6 = 2 \times 10^8$ FLOPs. On the same processor, this takes $2 \times 10^8/10^{11} = 0.002$ seconds, or **2 milliseconds**.

  The per-iteration speedup is also a factor of $10,000\times$. An optimization that might take a full day using a batch approach (assuming 8,640 iterations) could be completed in under 20 seconds with the streaming algorithm.

This is the practical payoff of the streamability test. It is not an incremental improvement; it is a phase transition in feasibility. It allows practitioners to identify which problems can be scaled to enormous sizes without requiring a corresponding explosion in computational resources. For problems in even higher dimensions ($d = 10^9$) or in resource-constrained environments (like edge devices), identifying streamability is the only way to make them tractable.

10. Algorithmic Verification Protocol

Algorithm 1: ULRA-Test for Streamability Verification
Input: Problem (R, D), desired rank K, confidence δ, threshold τ
Output: "Streamable" or "Not Streamable"
Procedure:

1. Set sample size $m \leftarrow CK\log(d/\delta)$ for universal constant C
2. For $j = 1$ to m:
   - Sample random point $\theta_j \in D$
   - Compute residual $r_j \leftarrow \theta_j - T(\theta_j)$
3. Form matrix $G \leftarrow [r_1, r_2, \ldots, r_m] \in \mathbb{R}^{d\times m}$
4. Compute SVD: $G = U\Sigma V^T$
5. Estimate approximation error: $\hat{\varepsilon}_K \leftarrow \|\Sigma_{K+1:}\|_F / \|\Sigma\|_F$
6. If $\hat{\varepsilon}_K \leq \tau$: Return "Streamable at rank K"
7. Else: Return "Not Streamable at rank K"

Theorem 5 (Verification Guarantees). *Algorithm 10 correctly identifies streamability with probability $1 - \delta$ when $m \geq CK\log(d/\delta)$ for universal constant C.*

11. Necessary and Sufficient Conditions

Theorem 6 (Necessary Conditions for Streamability). *If a problem is streamable at rank K, then:*

1. *The effective dimension of the residual space is at most K*
2. *For smooth problems, the Hessian has at most K significant eigenvalues near the optimum*
3. *The problem admits a finite-dimensional Koopman representation with contractive dynamics*

Theorem 7 (Sufficient Conditions via Separability). *If $R(\theta) = \sum_{j=1}^{m} f_j(\langle a_j, \theta\rangle)$ with $\text{rank}(\{a_j\}_{j=1}^m) \leq K$, then the problem is streamable at rank K with $\varepsilon_K = 0$.*

Proof. The gradient is $\nabla R(\theta) = \sum_{j=1}^{m} f_j'(\langle a_j, \theta\rangle)a_j$. Since $\text{rank}(\{a_j\}) \leq K$, we can write $a_j = \sum_{i=1}^{K} c_{ji}b_i$ for orthonormal $\{b_i\}$. This provides exact rank-K representation of the residual. □

 doi:10.20944/preprints202509.0955.v1

12. Impossibility Results

Conjecture 1 (Fundamental Limitations). *1.* ***Chaotic Systems****: The logistic map $x_{t+1} = 4x_t(1 - x_t)$ is non-learnable—no finite-rank streaming algorithm can approximate the dynamics with bounded error.*

2. ***Deep Networks****: For neural networks with $L \geq 2$ layers and generic weights, streaming algorithms cannot achieve batch-equivalent performance without exponential rank growth in network parameters.*

3. ***Discontinuous Functions****: Problems with discontinuous objectives admit no averaged operator, hence are non-learnable.*

Proof. (1) Chaotic dynamics have continuous Koopman spectrum—any finite approximation accumulates unbounded error.

(2) Deep networks exhibit gradient rank scaling with width$\times$depth, violating polynomial ULRA bounds.

(3) Discontinuity precludes the existence of nonexpansive operators required for averaging. □

13. Conclusions

We have established a complete mathematical hierarchy for optimization problems that serves as both rigorous theoretical framework and practical decision system:

1. **Rigorous Classification**: Theorem 1 provides exhaustive categorization with strict inclusions
2. **Computational Mappings**: Theorem 4 quantifies efficiency gains with concrete examples
3. **Koopman Integration**: Lemma 1 connects to dynamical systems theory
4. **Practical Verification**: Algorithm 10 enables streamability testing
5. **Fundamental Limits**: Theorem 1 establishes impossibility results

This framework resolves the fundamental question of when streaming algorithms can match batch performance while providing immediate guidance for computational efficiency optimization. The hierarchy unifies diverse optimization methods under rigorous theoretical foundations with explicit computational complexity mappings, enabling practitioners to make informed algorithmic choices based on problem structure and computational constraints.

Funding: None.

Data Availability Statement: No data were analyzed; all results are theoretical.

Acknowledgments: The author thanks the anonymous reviewers for their valuable feedback and suggestions that improved the clarity and rigor of this work.

Conflicts of Interest: The author declares no conflicts of interest.

Use of Artificial Intelligence: The author acknowledges the use of AI assistance in developing and refining the mathematical formulations and computational validations presented in this work. All theoretical results, proofs, and interpretations remain the responsibility of the author.

Appendix A. Canonical Examples

Appendix A.1. PCA via Oja's Algorithm (Exact Rank-1)

Consider $R(w) = -\langle w, \Sigma w\rangle$ subject to $\|w\| = 1$. Let $T(w) = \mathrm{Proj}_{\|w\|=1}\big(w + \eta\,(\Sigma w - \langle w, \Sigma w\rangle w)\big)$ with $\eta \in (0, \bar{\eta})$. Then T is α-averaged on the sphere for small enough η, and the residual $r(w) = w - T(w)$ admits the exact rank-1 form $r(w) = g_1(w)\,\langle h_1(w),\, w\rangle$ with $g_1(w) = \eta\,\Sigma w$ and $h_1(w) = w$. Hence $\varepsilon_1 = 0$ and PCA is streamable at rank 1.

Appendix A.2. Matrix Completion / Nuclear Norm via Frank–Wolfe

FW updates on $\min_X f(X) + \lambda\|X\|_*$ take $X_{t+1} = (1 - \gamma_t)X_t + \gamma_t\, u_t v_t^\top$ where (u_t, v_t) are top singular vectors of $\nabla f(X_t)$. Each step is a rank-1 atom; the residual is exactly rank-1 ($\varepsilon_1 = 0$).

 doi:10.20944/preprints202509.0955.v1

Under standard FW conditions (curvature constant, compact domain) the operator is averaged and convergent.

Appendix A.3. L1-Regularized Logistic Regression

Let $R(\theta) = \mathbb{E}[\log(1 + \exp(-y\langle x, \theta\rangle))] + \lambda\|\theta\|_1$. Define $T(\theta) = \mathrm{prox}_{\eta\lambda\|\cdot\|_1}(\theta - \eta\nabla f(\theta))$ for small enough η. Then T is α-averaged, and the prox is separable across coordinates, enabling block- or coordinate-streaming. If ∇f admits a low-rank factorization on D, the residual is rank-K with ε_K controlled by the factorization error.

Appendix A.4. Koopman EDMD

In EDMD coordinates, $z_{t+1} = Kz_t$ with $K \in \mathbb{R}^{m\times m}$. If $\rho(K) < 1$ in an induced norm, the linear operator is contractive (hence averaged), and batch iteration converges. If $\mathrm{rank}(K) \le K$ or K admits a uniform rank-K approximation on D, the residual is streamable with ε_K equal to the approximation error.

Appendix B. Recursive Streaming and Finite-Layer Networks

Proposition A1 (Recursive Streaming as Layered Composition)**.** *Let T be α-averaged on D and suppose the residual $R(\theta) = \theta - T(\theta)$ admits a rank-K factorization $R(\theta) \approx \sum_{i=1}^{K} g_i(\theta) \otimes h_i(\theta)^*$ with error ε_K. Consider T steps of the streaming iteration $\theta_{t+1} = \theta_t - \eta \sum_{i=1}^{K} g_i(\theta_t, \xi_t) \otimes h_i(\theta_t, \xi_t)^*$. Then $\theta_0 \mapsto \theta_T$ equals a depth-T composition of width-K residual blocks (with Lipschitz nonlinearity induced by the averaged map), i.e., a finite-layer neural architecture of width K and depth T, with approximation error accumulating as $O(T\,\varepsilon_K)$ under bounded Lipschitz constants.*

Appendix C. Real-Time Streaming Control Under Compute Budgets

Corollary A1 (Latency-Limited Streaming Control)**.** *Let a controller run N fixed-point iterations of T per control period Δt, with per-iteration budget $O(Kd)$. If the closed-loop residual admits a rank-K uniform approximation with error ε_K and T is contractive with modulus $\rho < 1$, then for sufficiently small step-size η the receding-horizon streaming controller stabilizes the system and attains steady-state tracking error $O(\varepsilon_K)$. Conversely, if the closed-loop problem is not streamable under the available budget, there is no guarantee that a fixed-iteration controller achieves stability or performance within Δt.*

References

1. H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces* (Springer, New York, 2011).
2. A. Beck, *First-Order Methods in Optimization* (SIAM, Philadelphia, 2017).
3. M. Jaggi, Revisiting Frank-Wolfe: Projection-free sparse convex optimization, in *Proceedings of the 30th International Conference on Machine Learning* (2013) pp. 427–435.
4. B. O. Koopman, Hamiltonian systems and transformation in Hilbert space, Proceedings of the National Academy of Sciences **17**, 315 (1931).
5. M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition, Journal of Nonlinear Science **25**, 1307 (2015).
6. E. Oja, Simplified neuron model as a principal component analyzer, Journal of Mathematical Biology **15**, 267 (1982).
7. A. Katharopoulos et al., Transformers are RNNs: Fast autoregressive transformers with linear attention, in *Proceedings of the 37th International Conference on Machine Learning* (2020) pp. 5156–5165.
8. T. Vogels et al., PowerSGD: Practical low-rank gradient compression for distributed optimization, in *Advances in Neural Information Processing Systems* (2019) pp. 14348–14358.