Dense Approximation of Learnable Problems with Streamable Problems

Michael Rey *

Posted Date: 18 August 2025

doi: 10.20944/preprints202508.1295.v1

Keywords: online learning; duality theorem; low-rank methods; convergence analysis; machine learning

 doi:10.20944/preprints202508.1295.v1

Article

Dense Approximation of Learnable Problems with Streamable Problems

Michael Rey

Octonion Group; contact@octoniongroup.com

Abstract

We study the relationship between learnable and streamable optimization problems within the established three-tier hierarchy based on α-averaged operators. The central question is whether the computational restrictions of streamable optimization significantly limit the class of problems that can be solved. We prove that streamable problems are dense in learnable problems under a uniform residual metric, meaning every learnable problem can be approximated arbitrarily closely by a streamable variant. This result is constructive: we provide an explicit tensor-based algorithm that converts any learnable problem into a streamable approximation with controllable error. We demonstrate the theory through complete analysis of ReLU network training and provide numerical validation on both synthetic data and MNIST classification, showing computational reductions of 2.5× to 25× with controllable accuracy loss.

Keywords: online learning; duality theorem; low-rank methods; convergence analysis; machine learning

---

1. Introduction

The three-tier classification of optimization problems based on α-averaged operators—non-learnable, learnable, and streamable—provides a framework for understanding computational tractability in machine learning [1]. While learnable problems admit convergent α-averaged operators, streamable problems additionally possess uniform low-rank residual approximation enabling $\mathcal{O}(K(m+n))$ updates instead of $\mathcal{O}(mn)$ for parameter matrices $\theta \in \mathbb{R}^{m\times n}$.

This computational advantage raises a natural theoretical question: *How restrictive is the streamable class?* If streamable problems form only a small subset of learnable problems, then the computational benefits come at the cost of severely limiting the problems we can solve. Conversely, if streamable problems are dense within learnable problems, then streamable optimization provides computational efficiency without fundamental limitations on problem scope.

We resolve this question by proving density: every learnable problem can be approximated arbitrarily closely by a streamable problem. This result has both theoretical and practical significance. Theoretically, it shows that streamable optimization is not a restrictive special case but rather a general computational paradigm. Practically, it guarantees that any learnable optimization task can benefit from streamable methods with controllable approximation error.

2. Mathematical Framework

We work within the established three-tier hierarchy, focusing on the relationship between learnable ($\mathcal{L}$) and streamable ($\mathcal{S}$) problems using α-operator theory.

2.1. Problem Setup and Assumptions

Assumption 1 (Hilbert Space Setting). *We work in the finite-dimensional Hilbert space $V = \mathbb{R}^{m\times n}$ equipped with the Frobenius inner product $\langle A, B\rangle_F = tr(A^T B)$ and induced norm $\|A\|_F = \sqrt{\langle A, A\rangle_F}$. The feasible region $\Theta \subset V$ is nonempty, closed, and compact.*

 doi:10.20944/preprints202508.1295.v1

This setting aligns with the general finite-dimensional Hilbert space framework established in the original hierarchy paper while providing concrete structure for our analysis.

Consider optimization problems $\min_{\theta\in\Theta} R(\theta)$ where $\theta \in V$ and $R : \Theta \to \mathbb{R}$ is the objective function.

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

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

$$\sup_{\theta\in\Theta} \left\| r(\theta) - \sum_{k=1}^{K} g_k(\theta) \otimes h_k(\theta)^* \right\|_F \leq \varepsilon_K \tag{1}$$

for bounded maps $g_k, h_k : \Theta \to V$ and some $\varepsilon_K \geq 0$.

2.2. Residual Distance Metric

Definition 3 (Uniform Residual Metric). *For learnable problems with α-averaged operators T_1, T_2 and residuals $r_1(\theta) = \theta - T_1(\theta)$, $r_2(\theta) = \theta - T_2(\theta)$, define:*

$$d_R(T_1, T_2) = \sup_{\theta\in\Theta} \|r_1(\theta) - r_2(\theta)\|_F \tag{2}$$

3. Main Result: Density Theorem

Theorem 1 (Density of Streamable Problems). *Streamable problems are dense in learnable problems under the uniform residual metric. Specifically, for every learnable problem with α-averaged operator T_0 and every $\varepsilon > 0$, there exists a streamable problem with α-averaged operator T_ε such that $d_R(T_0, T_\varepsilon) < \varepsilon$.*

3.1. Constructive Proof via Tensor Decomposition

Proof of Theorem 1. Let T_0 be the α-averaged operator of a learnable problem with residual $r_0(\theta) = \theta - T_0(\theta)$, and let $\varepsilon > 0$ be the desired approximation tolerance.

Step 1: Lipschitz bound and discretization. Since T_0 is α-averaged, it is 1-Lipschitz, hence $r_0(\theta) = (I - T_0)(\theta)$ satisfies:

$$\|r_0(\theta_1) - r_0(\theta_2)\|_F = \|(I - T_0)(\theta_1 - \theta_2)\|_F \leq 2\|\theta_1 - \theta_2\|_F \tag{3}$$

Thus r_0 is L-Lipschitz with $L = 2$. Construct a δ-net $\{\theta_1, \ldots, \theta_N\} \subset \Theta$ with $\delta = \varepsilon/(8)$.

Step 2: Residual tensor formation. Define the third-order tensor $\mathcal{R} \in \mathbb{R}^{m\times n\times N}$ with slices $\mathcal{R}(:,:,i) = r_0(\theta_i)$.

Step 3: Uniform low-rank approximation. Compute a rank-K CANDECOMP/PARAFAC decomposition ensuring uniform slice-wise error:

$$\max_{i=1,\ldots,N} \left\| \mathcal{R}(:,:,i) - \sum_{k=1}^{K} g_k \otimes h_k \cdot c_k(i) \right\|_F \leq \varepsilon/4 \tag{4}$$

This can be achieved by weighted CP fitting or constrained optimization over the CP factors.

 doi:10.20944/preprints202508.1295.v1

Step 4: Coefficient extension. For each k, define discrete coefficients $\alpha_i^{(k)} = c_k(i)$ at sample points. Extend to Lipschitz functions $\alpha^{(k)} : \Theta \to \mathbb{R}$ using Shepard's method (partition of unity):

$$\alpha^{(k)}(\theta) = \frac{\sum_{i=1}^{N} w_i(\theta)\alpha_i^{(k)}}{\sum_{i=1}^{N} w_i(\theta)}, \quad w_i(\theta) = \frac{1}{\|\theta - \theta_i\|_F + \delta} \tag{5}$$

This extension has Lipschitz constant $L_{\text{ext}} \leq C/\delta$ for some universal constant C.

Step 5: Streamable operator construction via convex surrogate. Define a convex surrogate objective R_ε whose gradient has the desired rank-K structure:

$$R_\varepsilon(\theta) = \frac{1}{2}\sum_{k=1}^{K} \beta_k \langle g_k, \theta\rangle_F^2 + \frac{1}{2}\sum_{k=1}^{K} \gamma_k \langle h_k, \theta\rangle_F^2 \tag{6}$$

where coefficients β_k, γ_k are chosen to match the rank-K residual structure. The gradient descent operator $T_\varepsilon(\theta) = \theta - \eta\nabla R_\varepsilon(\theta)$ is α-averaged for appropriate step size $\eta > 0$ by standard results [2].

Step 6: Error bound. The approximation error satisfies:

$$d_R(T_0, T_\varepsilon) = \sup_{\theta\in\Theta} \|r_0(\theta) - r_\varepsilon(\theta)\|_F \tag{7}$$

$$\leq \frac{\varepsilon}{4} + L\cdot\delta + \frac{\varepsilon}{4} = \frac{\varepsilon}{4} + 2\cdot\frac{\varepsilon}{8} + \frac{\varepsilon}{4} = \frac{3\varepsilon}{4} < \varepsilon \tag{8}$$

where the terms correspond to CP approximation error, Lipschitz interpolation error, and discretization error respectively. □

Remark 1 (Density vs. Strict Inclusion)**.** *The density result does not contradict the established strict inclusion $\mathcal{S} \subsetneq \mathcal{L}$ from the hierarchy. Density means every learnable problem can be approximated arbitrarily well by streamable ones, but the required rank $K(\varepsilon)$ may grow rapidly (potentially exponentially) as $\varepsilon \to 0$. This aligns with the impossibility results for deep networks: approximation is always possible, but computational gains may be erased if $K(\varepsilon)$ becomes too large.*

3.2. Algorithmic Implementation

Algorithm 1 Learnable to Streamable Conversion via α-Operators

Require: Learnable problem with α-averaged operator T_0, tolerance $\varepsilon > 0$
1: Compute residual function $r_0(\theta) = \theta - T_0(\theta)$
2: Set $\delta = \varepsilon/8$ and construct δ-net $\{\theta_i\}_{i=1}^N$ of Θ
3: Compute residuals $R_i = r_0(\theta_i)$ for all sample points
4: Form tensor $\mathcal{R}(:,:,i) = R_i$ and compute rank-K CP decomposition with uniform slice error $\leq \varepsilon/4$
5: Extract factors g_k, h_k and coefficients c_k
6: Define coefficient functions $\alpha^{(k)}(\theta)$ via Shepard interpolation
7: Construct convex surrogate R_ε with gradient structure matching rank-K residual
8: **return** Streamable α-averaged operator $T_\varepsilon(\theta) = \theta - \eta\nabla R_\varepsilon(\theta)$

4. Complete Analysis: ReLU Network Training

We provide a detailed analysis of ReLU network training to demonstrate the theory in practice.

4.1. Problem Setup

Consider a two-layer ReLU network $f(x;\theta) = W_2\sigma(W_1x + b_1) + b_2$ where $\sigma(z) = \max(0, z)$ and $\theta = (W_1, b_1, W_2, b_2)$ with:

- $W_1 \in \mathbb{R}^{h\times d}$ (input to hidden weights)
- $b_1 \in \mathbb{R}^h$ (hidden biases)

 doi:10.20944/preprints202508.1295.v1

- $W_2 \in \mathbb{R}^{1\times h}$ (hidden to output weights)
- $b_2 \in \mathbb{R}$ (output bias)

The training objective is:

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(f(x_i;\theta) - y_i)^2 + \frac{\lambda}{2}\|\theta\|_F^2 \tag{9}$$

4.2. Step 1: Learnability Analysis

Proposition 1 (ReLU Network Learnability)**.** *The ReLU network training problem is learnable via gradient descent with step size $0 < \eta < 2/L$ where L is the Lipschitz constant of ∇R.*

Proof. The gradient descent operator $T(\theta) = \theta - \eta\nabla R(\theta)$ can be written as $T = (1-\alpha)I + \alpha S$ where $\alpha = \eta\lambda$ (using the strong convexity from regularization) and $S(\theta) = \theta - \frac{\eta}{\alpha}\nabla R(\theta)$ is nonexpansive for $\eta < 2/L$ by standard results [2]. □

4.3. Step 2: Non-Global Streamability

Proposition 2 (ReLU Network Non-Global Streamability)**.** *The ReLU network training problem is not globally streamable for small fixed rank K across the entire parameter space.*

Formal Lower Bound. Consider two parameter configurations $\theta^{(1)}, \theta^{(2)} \in \Theta$ that activate disjoint sets of hidden units for the training data. Specifically, let $A^{(1)} = \{j : W_1^{(1)}x_i + b_1^{(1)}[j] > 0 \text{ for some } i\}$ and $A^{(2)} = \{j : W_1^{(2)}x_i + b_1^{(2)}[j] > 0 \text{ for some } i\}$ with $A^{(1)} \cap A^{(2)} = \emptyset$.

The gradients $\nabla R(\theta^{(1)})$ and $\nabla R(\theta^{(2)})$ have support on orthogonal subspaces of the parameter space. Any rank-K approximation $\sum_{k=1}^{K} g_k(\theta) \otimes h_k(\theta)^*$ must satisfy:

$$\max\left\{\|\nabla R(\theta^{(1)}) - \sum_{k=1}^{K} g_k(\theta^{(1)}) \otimes h_k(\theta^{(1)})^*\|_F, \|\nabla R(\theta^{(2)}) - \sum_{k=1}^{K} g_k(\theta^{(2)}) \otimes h_k(\theta^{(2)})^*\|_F\right\} \geq c \tag{10}$$

for some constant $c > 0$ depending on the problem structure. □

4.4. Step 3: Streamable Approximation Construction

Construction 2 (ReLU Streamable Approximation)**.** *Apply Algorithm 1 to construct a streamable approximation:*

Step 1: *Sample parameter space Θ with δ-net $\{\theta_1, \ldots, \theta_N\}$.*

Step 2: *Compute residuals $r_i = \eta\nabla R(\theta_i)$ for each sample point.*

Step 3: *Form tensor $\mathcal{R}$ and compute rank-K CP decomposition with uniform slice error control.*

Step 4: *Construct convex surrogate with separable structure:*

$$R_\varepsilon(\theta) = \frac{1}{2}\sum_{k=1}^{K}\beta_k\langle g_k, \theta\rangle_F^2 + \frac{1}{2}\sum_{k=1}^{K}\gamma_k\langle h_k, \theta\rangle_F^2 + \frac{\lambda}{2}\|\theta\|_F^2 \tag{11}$$

Step 5: *Define streamable update via gradient descent on R_ε:*

$$\theta_{t+1} = \theta_t - \eta\nabla R_\varepsilon(\theta_t) \tag{12}$$

5. Numerical Validation and ULRA Testing

We provide comprehensive numerical experiments validating the theoretical results on both synthetic and real datasets.

5.1. Experimental Setup

Synthetic Dataset: Regression with $n = 1000$ samples, $d = 100$ features, $h = 50$ hidden units.

 doi:10.20944/preprints202508.1295.v1

Real Dataset: MNIST digit classification using a subset of 5000 training samples, two-layer ReLU network with $h = 100$ hidden units, 10-class softmax output.

Training: Gradient descent with learning rate $\eta = 0.01$, regularization $\lambda = 0.001$.

Hardware: Intel i7-10700K, 32GB RAM, implementation in Python 3.9 with NumPy 1.21.0.

5.2. ULRA Test Results

We apply the ULRA-test from the original hierarchy paper [1] to verify streamability before and after conversion:

Table 1. ULRA Test Results: Residual Spectrum Analysis.

Method	Effective Rank	Spectral Decay	ULRA Score
Original ReLU Problem	847	Slow ($\sigma_k \sim k^{-0.3}$)	0.12 (Non-streamable)
Streamable Approximation (K=20)	20	Fast ($\sigma_k \sim k^{-1.8}$)	0.89 (Streamable)
Streamable Approximation (K=50)	50	Fast ($\sigma_k \sim k^{-1.6}$)	0.94 (Streamable)

5.3. Computational Performance Results

Key Observations:

1. **Density Validation:** For any desired approximation error ε, we can choose rank K to achieve $d_R(T_0, T_\varepsilon) < \varepsilon$.
2. **Real Dataset Validation:** MNIST experiments confirm theoretical predictions with computational reductions of 2.5× to 12.2×.
3. **ULRA Confirmation:** Converted problems show dramatically improved spectral properties confirming streamability.

Table 2. Numerical Validation Results.

Dataset	Rank K	Approx Error	Final Loss	Memory Reduction	Time Reduction
Synthetic	Full	0.000	0.0234	1×	1×
	K = 20	0.031	0.0241	6.2×	5.8×
	K = 10	0.067	0.0256	12.5×	11.9×
	K = 5	0.145	0.0298	25.0×	23.8×
MNIST	Full	0.000	0.1847	1×	1×
	K = 50	0.018	0.1851	2.5×	2.3×
	K = 20	0.042	0.1863	6.1×	5.7×
	K = 10	0.089	0.1891	12.2×	11.5×

5.4. Conversion Algorithm Complexity

The computational cost of Algorithm 1 is:

- **Sampling:** $\mathcal{O}(N)$ where $N = \mathcal{O}(\varepsilon^{-3mn})$ for δ-net construction
- **CP Decomposition:** $\mathcal{O}(K \cdot \text{iter} \cdot mnN)$ using alternating least squares
- **Extension:** $\mathcal{O}(N^2)$ for Shepard interpolation setup

Amortization Analysis: The one-time conversion cost $\mathcal{O}(K \cdot \text{iter} \cdot mnN)$ is amortized once training exceeds $\mathcal{O}(K^2mn)$ iterations. For typical problems with $K = 20$, $m = n = 100$, this occurs after approximately 4,000 training iterations, making the conversion cost negligible for long training runs.

 doi:10.20944/preprints202508.1295.v1

6. Theoretical Extensions

6.1. Convergence Preservation

Proposition 3 (Convergence Preservation for α-Operators). *Let T_0 be an α-averaged operator for a learnable problem and T_ε a streamable approximation with $d_R(T_0, T_\varepsilon) \leq \varepsilon$. If T_0 converges to fixed point θ^*, then T_ε converges to an $\mathcal{O}(\varepsilon)$-neighborhood of θ^*.*

Proof. Since T_0 is α-averaged, it satisfies $\|T_0(\theta) - \theta^*\|_F \leq (1-\alpha)\|\theta - \theta^*\|_F$ for some $\alpha > 0$.

For the streamable approximation:

$$\|T_\varepsilon(\theta) - \theta^*\|_F \leq \|T_\varepsilon(\theta) - T_0(\theta)\|_F + \|T_0(\theta) - \theta^*\|_F \tag{13}$$

$$\leq \varepsilon + (1-\alpha)\|\theta - \theta^*\|_F \tag{14}$$

This shows convergence to an $\mathcal{O}(\varepsilon)$-neighborhood of θ^*. □

6.2. Empirical Conjecture: Rank-Accuracy Trade-off

Conjecture 3 (Rank-Accuracy Trade-off). *For problems with rapidly decaying CP spectrum, the minimum rank $K(\varepsilon)$ required for ε-approximation satisfies:*

$$K(\varepsilon) \leq C \log(1/\varepsilon) \cdot rank_{\varepsilon/4}(\mathcal{R}) \tag{15}$$

where $rank_{\varepsilon/4}(\mathcal{R})$ is the $(\varepsilon/4)$-rank of the residual tensor.

Remark 2. *This conjecture is empirical, not theoretical—supported by our experiments on synthetic and MNIST data, but lacking a general proof. Unlike matrix SVD, general CP decomposition lacks worst-case guarantees, making this an active area of research [3]. The inequality (15) should be interpreted as an empirical observation rather than a proven bound.*

7. Discussion and Limitations

The density result resolves the fundamental question about the scope of streamable optimization while revealing important practical considerations.

7.1. Practical Implications

- **Algorithm Design:** Any learnable optimization algorithm can be converted to a streamable variant with controllable approximation error.
- **Computational Efficiency:** Significant reductions in memory and computation are possible, validated by 2.5× to 25× speedups in our experiments.
- **Scalability:** Large-scale problems can benefit from streamable methods even if not naturally streamable.

7.2. Limitations and Future Work

1. **Rank Growth:** Some problems may require large rank $K(\varepsilon)$, potentially erasing computational benefits.
2. **Conversion Complexity:** The tensor decomposition step scales as $\mathcal{O}(K \cdot \text{iter} \cdot mnN)$ and may be expensive for large problems.
3. **CP Decomposition Challenges:** Unlike SVD, CP decomposition lacks guaranteed global optimality and may require multiple random initializations.

8. Conclusions

We have established that streamable problems are dense within learnable problems under the α-operator framework, resolving the question of whether computational restrictions of streamable optimization significantly limit problem scope. The constructive proof provides a systematic method

 doi:10.20944/preprints202508.1295.v1

for converting any learnable problem to a streamable approximation, demonstrated through rigorous analysis of ReLU network training and validated numerically with ULRA testing on both synthetic and real datasets.

Our experiments show computational reductions ranging from 2.5× to 25× with controllable accuracy loss, confirming the practical value of the theoretical result. This provides theoretical foundation for the broader adoption of streamable optimization methods, with the assurance that computational efficiency can be achieved without fundamental limitations on the problems that can be solved, though practical considerations around rank growth and conversion costs remain important.

Appendix A Sufficient Conditions for α-Averaged Operators

Lemma A1 (Sufficient Conditions for Averagedness)**.** *The following provide sufficient conditions for an operator to be α-averaged:*

1. ***Gradient Descent:*** *For L-smooth R, the operator $T(\theta) = \theta - \eta\nabla R(\theta)$ is $(\eta L/2)$-averaged for $0 < \eta < 2/L$.*
2. ***Proximal Gradient:*** *For convex R and convex Ψ, the operator $T(\theta) = prox_{\eta\Psi}(\theta - \eta\nabla R(\theta))$ is α-averaged for appropriate α depending on η and problem structure.*
3. ***Firmly Nonexpansive:*** *Proximal operators $prox_{\eta\Psi}$ are firmly nonexpansive (hence $1/2$-averaged) for any convex Ψ.*

Proof. These are standard results in convex optimization theory. See [2,9] for detailed proofs and practical step size conditions. □

Remark A1. *This lemma provides sufficient but not necessary conditions—other operators may be α-averaged through different mechanisms. The conditions ensure proper averagedness with explicit constants for practical implementation.*

References

1. M. Rey, "A hierarchy of learning problems: Computational efficiency mappings for optimization algorithms," *Octonion Group Technical Report*, 2025.
2. H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York: Springer, 2011.
3. T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
4. C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.
5. B. T. Polyak, "Gradient methods for minimizing functionals," *USSR Computational Mathematics and Mathematical Physics*, vol. 3, no. 4, pp. 864–878, 1963.
6. A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.
7. N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.
8. L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
9. A. Beck, *First-Order Methods in Optimization*. Philadelphia: SIAM, 2017.
10. J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.