AIether: Procedural Growth of Neural Networks via Adaptive Geometric Extrapolation

Samuel Rocha

February 2026

Abstract

We present **AIether**, a procedural growth system for deep neural networks based on adaptive geometric extrapolation. The method uses spectral analysis of the optimization trajectory to construct informed initializations for new layers, preserving geometric properties of the training dynamics. This document details: (1) geometric stagnation metrics (τ, κ); (2) dynamic subspace decomposition via SVD; (3) tensor-wise multi-scale extrapolation; (4) adaptive criteria for architectural expansion. The approach is mathematically rigorous and computationally efficient, suitable for large-scale models.

Contents

1 Introduction

Procedural growth of neural architectures faces the fundamental challenge of initializing new layers such that: (1) continuity of the learned function is preserved; (2) the parameter space is explored efficiently; (3) stagnation regions are detected and escaped.

Traditional methods (random initialization, layer copying) ignore the rich geometric structure of the optimization trajectory. AIether proposes an approach based on *adaptive geometric extrapolation*, which analyzes the temporal history of parameters to construct informed initializations.

1.1 Motivation

During training of deep neural networks, parameters trace a complex trajectory in weight space. This trajectory contains valuable information about:

- **Productive directions**: Subspaces where the optimizer makes consistent progress
- **Stagnation regions**: Areas where gradients are small or oscillate excessively
- **Spectral structure**: Dominant modes of variation that capture learning patterns

When adding a new layer, we want to leverage this historical knowledge to:

1. Position the layer in a promising region of parameter space
2. Avoid re-learning structures already captured
3. Facilitate continuity of the represented function

1.2 Main Contributions

1. **Geometric formalization**: Rigorous characterization of the optimization trajectory as a curve in tensor spaces, with quantitative quality metrics.
2. **Tensor-wise extrapolation**: Algorithm that operates individually on each parameter tensor, respecting their specific geometric structures (2D matrices vs. 1D vectors).
3. **Multi-scale adaptation**: Adaptive extrapolation coefficients based on global (τ), local (κ), and spectral (ρ) metrics.
4. **Expansion criteria**: Quantitative conditions to detect stagnation and trigger architectural growth.

2 Geometry of the Optimization Trajectory

2.1 State Space and Discrete Trajectory

Definition 1 (Parameter Space). *Let $\mathcal{W} = \mathbb{R}^{m \times n}$ be the space of weight matrices for a layer. For layers with multiple tensors (attention, MLP, normalization), the parameter space is the Cartesian product:*

$$\mathcal{P} = \mathcal{W}_1 \times \mathcal{W}_2 \times \cdots \times \mathcal{W}_K,$$

where each $\mathcal{W}_i$ can be a matrix $\mathbb{R}^{m_i \times n_i}$ or vector $\mathbb{R}^{d_i}$.

Definition 2 (Optimization Trajectory). *A sequence of checkpoints during training defines a discrete trajectory:*

$$\mathcal{T} = \{W_0, W_1, \ldots, W_{S-1}\} \subset \mathcal{W},$$

where W_t represents the parameter state at temporal step t, sampled every Δt optimization iterations.

Remark 1 (Geometric Interpretation). *The trajectory $\mathcal{T}$ can be viewed as a discrete sampling of a continuous curve in parameter space. Our analysis seeks to infer properties of this underlying curve: instantaneous velocity, local curvature, principal directions of variation, and global behavior (efficiency, tortuosity).*

2.2 Fundamental Geometric Metrics

2.3 Discrete Velocity and Acceleration

The temporal evolution of parameters can be characterized through discrete approximations of first and second-order derivatives, analogous to forward and central difference schemes in numerical analysis.

Definition 3 (Discrete Velocity and Acceleration). *The velocity between consecutive checkpoints is defined as:*

$$V_t := W_t - W_{t-1} \in \mathcal{W}, \quad t = 1, \ldots, S-1,$$

with mean velocity:

$$\bar{V} := \frac{1}{S-1} \sum_{t=1}^{S-1} V_t = \frac{\Delta}{S-1},$$

where $\Delta := W_{S-1} - W_0$ denotes the total displacement.

The acceleration, capturing changes in velocity magnitude and direction, is:

$$A_t := V_{t+1} - V_t = W_{t+1} - 2W_t + W_{t-1}, \quad t = 1, \ldots, S-2,$$

with mean acceleration:

$$\bar{A} := \frac{1}{S-2} \sum_{t=1}^{S-2} A_t.$$

These discrete operators correspond to finite difference approximations of continuous derivatives: V_t approximates $\frac{dW}{dt}$ at the midpoint between steps, while A_t approximates $\frac{d^2W}{dt^2}$ through a central difference scheme. The velocity field $\{V_t\}$ encodes the trajectory's dominant direction and rate of parameter evolution, while the acceleration field $\{A_t\}$ reveals trajectory curvature and directional instability. Persistent high-magnitude acceleration indicates frequent directional changes, suggesting the optimizer encounters complex landscape geometry or lacks consistent gradient signal. Both quantities serve as fundamental inputs to the extrapolation formula (Section 3), where $\bar{V}$ provides the linear trend component and $\bar{A}$ enables second-order trajectory prediction.

2.4 Temporal Stretch Factor (TSF)

Quantifying trajectory efficiency requires comparing the actual path traversed against the most direct route between endpoints. This motivates the definition of a normalized path length metric.

Definition 4 (Path Length and TSF)**.** *The total arc length traveled by the trajectory is:*

$$\mathcal{L}_{path} := \sum_{t=1}^{S-1} \|V_t\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The Euclidean distance between endpoints is:

$$\mathcal{L}_{direct} := \|\Delta\|_F = \|W_{S-1} - W_0\|_F.$$

*The **Temporal Stretch Factor** (TSF) is defined as:*

$$\boxed{\tau := \frac{\mathcal{L}_{path}}{\mathcal{L}_{direct} + \varepsilon}}$$

where $\varepsilon > 0$ is a small regularization constant for numerical stability.

This metric bears conceptual similarity to trajectory tortuosity indices employed in movement ecology [4] and arc-length parametrization in differential geometry [7], though adapted specifically for discrete parameter trajectories in optimization contexts.

Proposition 1 (Properties of TSF)**.** *The factor τ satisfies:*

1. ***Lower bound**: $\tau \geq 1$ always (by the triangle inequality), with equality if and only if the trajectory is a line segment.*

2. ***Waste quantification**: The excess $\tau - 1$ measures the fraction of movement attributable to oscillations, backtracking, or lateral exploration relative to net displacement.*

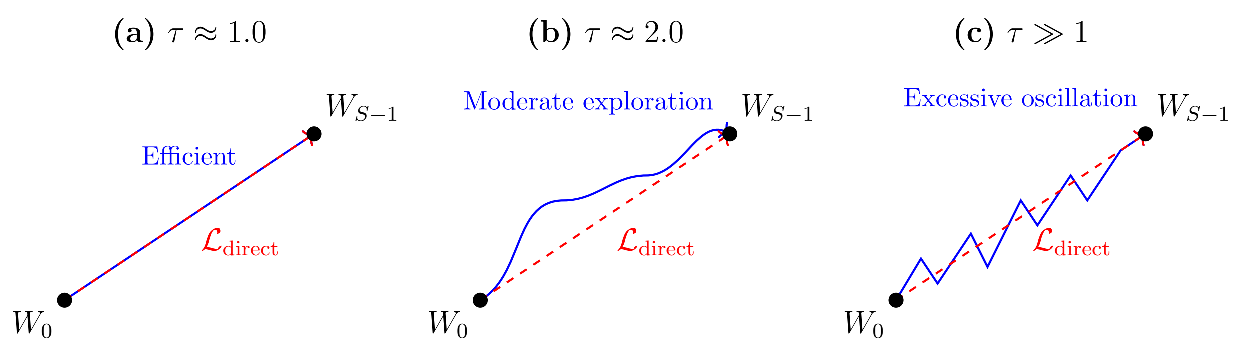

Figure 1: Geometric interpretation of the Temporal Stretch Factor (τ). The ratio between cumulative path length (blue trajectory) and direct displacement (red dashed line) quantifies optimization efficiency. (a) Nearly rectilinear movement indicates efficient convergence. (b) Moderate curvature reflects healthy exploration of the loss landscape. (c) Chaotic oscillation suggests stagnation or gradient instability.

In preliminary experiments on language model pretraining using subsets of FineWebEdu with models in the 10M–50M parameter range, observed values of τ computed over sliding windows of $S = 5$–10 checkpoints exhibited the following empirical patterns: values near unity ($\tau \approx 1.0$–1.2) corresponded to rapid convergence phases, moderate values ($\tau \approx 1.5$–2.5) characterized stable training with healthy exploration, while elevated values ($\tau > 3.0$) consistently preceded training plateaus or required learning rate adjustments. These observations inform the stagnation detection thresholds discussed in Section 4.

2.5 Effective Curvature

While the temporal stretch factor τ quantifies global trajectory efficiency, it does not distinguish between smooth curved paths and erratic oscillatory movement. A complementary local metric is required to detect situations where the optimizer makes substantial movement yet achieves minimal net progress toward the endpoint.

Definition 5 (Parallel-Perpendicular Decomposition)**.** *For each velocity increment V_t, we perform an orthogonal decomposition relative to the global displacement direction $\hat{\Delta} := \Delta/\|\Delta\|_F$:*

$$V_t = V_t^{\parallel} + V_t^{\perp},$$

where:

$$V_t^{\parallel} := \langle V_t, \hat{\Delta} \rangle_F \cdot \hat{\Delta}, \tag{1}$$

$$V_t^{\perp} := V_t - V_t^{\parallel}, \tag{2}$$

and $\langle A, B \rangle_F := \mathrm{Tr}(A^\top B)$ denotes the Frobenius inner product. The component $V_t^{\parallel}$ represents useful progress along the net displacement direction, while $V_t^{\perp}$ captures lateral deviation.

Definition 6 (Effective Curvature)**.** *The* ***effective curvature*** *quantifies trajectory tortuosity as the ratio of accumulated lateral deviation to net directional progress:*

$$\boxed{\kappa := \frac{\sum_{t=1}^{S-1} \|V_t^{\perp}\|_F}{\sum_{t=1}^{S-1} \|V_t^{\parallel}\|_F + \varepsilon}}$$

This metric provides a computationally efficient proxy for trajectory curvature without requiring Hessian computation or second-order gradient information. The construction parallels the notion of geodesic curvature in differential geometry [7], adapted to discrete trajectories in high-dimensional parameter spaces. The metric is dimensionless and invariant under uniform rescaling of parameters, depending only on the geometric configuration of the trajectory.

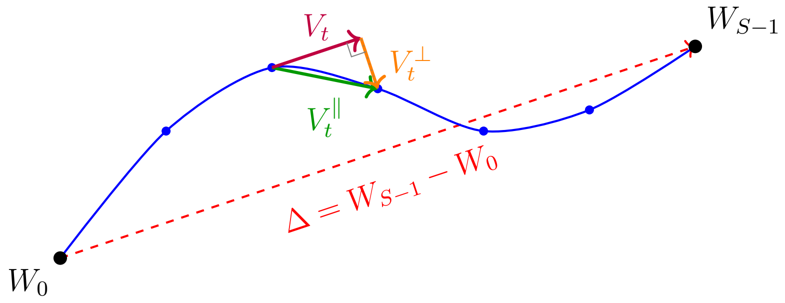

Figure 2: Geometric interpretation of effective curvature (κ). Each velocity increment V_t (purple) is decomposed into components parallel ($V_t^{\parallel}$, green) and perpendicular ($V_t^{\perp}$, orange) to the global displacement Δ (red dashed line). The metric captures the ratio of accumulated lateral deviation to net progress, serving as a proxy for trajectory tortuosity.

In preliminary experiments on language model pretraining using subsets of FineWebEdu with models in the 10M–50M parameter range under the same checkpoint regime ($S = 5$–10), observed values of κ exhibited characteristic patterns correlated with optimization behavior. Values near zero ($\kappa \approx 0$–0.1) indicated nearly rectilinear trajectories during efficient convergence phases. Moderate values ($\kappa \approx 0.3$–0.8) characterized stable training with balanced exploration of the loss landscape. Elevated values ($\kappa > 1.0$) consistently preceded or coincided with training stagnation, indicating movement predominantly orthogonal to net progress—the optimizer "wandering" without consistent directional improvement.

The effective curvature serves as a local complement to the global stretch factor τ. While both metrics increase during poor optimization, they capture distinct geometric pathologies: τ detects excessive total path length regardless of smoothness, whereas κ specifically identifies lateral oscillation relative to net displacement. In practice, stagnation detection employs both metrics conjunctively (Section 4), requiring simultaneous elevation of τ and κ to trigger architectural expansion, thereby reducing false positives from transient exploratory phases.

2.6 Decomposition into Dynamic Subspaces

Beyond quantifying trajectory quality through global (τ) and local (κ) metrics, effective extrapolation requires identifying the **intrinsic directions** along which the optimizer makes systematic progress. This motivates a spectral decomposition of the velocity field to extract low-dimensional structure from the high-dimensional parameter trajectory.

2.6.1 Velocity Matrix Construction

The temporal sequence of parameter changes is organized into a structured format amenable to linear algebraic analysis.

Definition 7 (Centered Velocity Matrix)**.** *Let $V_t := W_t - W_{t-1}$ denote the discrete velocity at step t for $t = 1, \ldots, S-1$. Define the mean velocity:*

$$\bar{V} := \frac{1}{S-1} \sum_{t=1}^{S-1} V_t = \frac{W_{S-1} - W_0}{S-1}.$$

For matrices $W_t \in \mathbb{R}^{m \times n}$, vectorize each velocity as $v_t := \operatorname{vec}(V_t) \in \mathbb{R}^D$ where $D = mn$. The ***centered velocity matrix*** *is:*

$$\tilde{V} := \begin{bmatrix} (v_1 - \bar{v})^\top \\ (v_2 - \bar{v})^\top \\ \vdots \\ (v_{S-1} - \bar{v})^\top \end{bmatrix} \in \mathbb{R}^{(S-1) \times D},$$

where $\bar{v} := \operatorname{vec}(\bar{V})$ denotes the vectorized mean velocity.

Remark 2 (Importance of Centering)**.** *Centering the velocity field is critical for isolating* ***variations around the dominant trend*** *rather than the trend itself. Without centering, the first principal component would simply recapture $\bar{V}$ (the net displacement direction), providing no information about oscillatory or exploratory behavior. Centered SVD reveals the structure of deviations from the mean trajectory, which is precisely what informs adaptive extrapolation and stagnation detection.*

2.6.2 Singular Value Decomposition

Spectral analysis via SVD identifies the principal axes of variation in the centered velocity field.

Definition 8 (Dynamic Subspace via SVD)**.** *The singular value decomposition of $\tilde{V}$ yields:*

$$\tilde{V} = U \Sigma Q^\top,$$

where:

- *$U \in \mathbb{R}^{(S-1) \times r}$ contains left singular vectors (temporal patterns),*
- *$\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$,*
- *$Q = [q_1, \ldots, q_r] \in \mathbb{R}^{D \times r}$ contains right singular vectors (spatial patterns),*

- $r := \operatorname{rank}(\tilde{V}) \leq \min(S-1, D)$ *denotes the effective rank.*

The ***dynamic subspace of dimension*** k *is defined as:*

$$\mathcal{U}_k := \operatorname{span}\{q_1, \dots, q_k\} \subset \mathbb{R}^D,$$

with corresponding orthogonal projection operator:

$$P_k := Q_k Q_k^\top, \quad \textit{where} \quad Q_k := [q_1, \dots, q_k] \in \mathbb{R}^{D \times k}.$$

Proposition 2 (Optimality of SVD Projection)**.** *Among all k-dimensional subspaces $\mathcal{U} \subset \mathbb{R}^D$, the subspace $\mathcal{U}_k$ minimizes the mean squared reconstruction error of the centered velocities:*

$$\mathcal{U}_k = \underset{\substack{\mathcal{U} \subset \mathbb{R}^D \\ \dim(\mathcal{U}) = k}}{\operatorname{argmin}} \frac{1}{S-1} \sum_{t=1}^{S-1} \|v_t - \bar{v} - P_{\mathcal{U}}(v_t - \bar{v})\|^2,$$

where $P_{\mathcal{U}}$ denotes orthogonal projection onto $\mathcal{U}$. This follows directly from the Eckart-Young-Mirsky theorem [2].

Remark 3 (Geometric Interpretation of Principal Modes)**.** *The right singular vectors $\{q_i\}_{i=1}^k$ represent the* ***principal directions of velocity variation*** *in parameter space. Specifically:*

- *q_1 captures the direction of maximum variance in centered velocities (the dominant oscillatory or exploratory mode),*
- *$q_2, \dots, q_k$ capture successively weaker modes of systematic variation,*
- *The singular values $\{\sigma_i\}$ quantify the magnitude of variation along each mode.*

Recent work on neural network optimization trajectories [16, 5] has shown that gradient descent typically explores a low-dimensional subspace of the full parameter space, with the effective dimensionality often much smaller than the nominal parameter count. The dynamic subspace $\mathcal{U}_k$ provides an empirical estimate of this intrinsic dimensionality over the recent optimization history.

Remark 4 (Choice of Subspace Dimension k)**.** *The dimension k controls a fundamental trade-off:*

- ***Small*** *k ($k \ll r$): Extrapolation is confined to a few dominant modes, reducing noise sensitivity but potentially discarding informative secondary structure. This is conservative and suitable for noisy or erratic trajectories.*
- ***Large*** *k ($k \approx r$): Retains fine-grained variation patterns, enabling more faithful trajectory reproduction but at the risk of overfitting to transient fluctuations.*

A practical heuristic is to select k such that the first k modes capture a target fraction (e.g., 85%–95%) of the total variance:

$$\frac{\sum_{i=1}^k \sigma_i^2}{\sum_{i=1}^r \sigma_i^2} \geq \theta_{var},$$

though the default fixed choice $k \in [5, 10]$ proves robust across diverse settings in preliminary experiments.

2.6.3 Secondary Modes and Escape Directions

While the primary subspace $\mathcal{U}_k$ captures dominant trajectory patterns, the secondary modes $\{q_{k+1}, \ldots, q_{k+\ell}\}$ play a crucial role in constructing exploration directions orthogonal to the main trajectory.

Definition 9 (Spectral Escape Direction)**.** *When the trajectory exhibits strong confinement to $\mathcal{U}_k$ (i.e., $\|(I - P_k)\bar{v}\| \approx 0$), an alternative exploration direction is constructed from weighted secondary modes:*

$$v_{escape} := \frac{\sum_{i=k+1}^{\min(k+\ell,r)} \sigma_i \cdot q_i}{\left\|\sum_{i=k+1}^{\min(k+\ell,r)} \sigma_i \cdot q_i\right\| + \varepsilon},$$

where $\ell \in [3, 10]$ controls the number of secondary modes incorporated, and $\varepsilon > 0$ ensures numerical stability.

Remark 5 (Rationale for Weighted Combination)**.** *The weighting by singular values σ_i prioritizes secondary directions with larger systematic variation over those dominated by noise. This construction provides a principled escape mechanism that:*

- *Remains orthogonal to $\mathcal{U}_k$ by construction,*
- *Leverages systematic (though subdominant) patterns in the trajectory,*
- *Avoids completely random exploration, which could destabilize training.*

This spectral escape mechanism will be incorporated into the extrapolation formula (Section 3) to facilitate navigation away from stagnation regions while maintaining geometric coherence with the optimization history.

Remark 6 (Degenerate Cases)**.** *In practice, several edge cases require special handling:*

1. ***Low effective rank*** *($r \ll D$): When the trajectory is highly confined to a low-dimensional manifold, $\mathcal{U}_k$ may capture essentially all variation. This is typical in later training stages and suggests the optimizer has converged to a low-dimensional attractor [14].*
2. ***Numerical rank deficiency****: If $\sigma_k/\sigma_1 < \varepsilon_{tol}$ (e.g., $\varepsilon_{tol} = 10^{-6}$), modes beyond index k contain primarily numerical noise and should be discarded. The effective subspace dimension should be truncated accordingly.*
3. ***Insufficient history*** *($S - 1 < k$): When fewer checkpoints are available than the desired subspace dimension, set $k \leftarrow \min(k, S - 1)$ automatically. This ensures $\mathcal{U}_k$ is well-defined though potentially information-limited.*

The implementation (Algorithm 1) handles these cases through adaptive truncation and regularization.

2.6.4 Computational Considerations

For large-scale models where $D = mn$ may exceed 10^7 parameters, computing the full SVD of $\tilde{V} \in \mathbb{R}^{(S-1)\times D}$ can be prohibitively expensive. Fortunately, since we require only the first $k + \ell$ components (typically $k + \ell \ll \min(S - 1, D)$), truncated SVD algorithms offer substantial computational savings.

Remark 7 (Randomized and Truncated SVD)**.** *Modern implementations employ randomized algorithms [6] that compute the top-k singular vectors in $O(SDk)$ time rather than $O(SD^2)$ for full SVD. For a 7B parameter model with $S = 10$ and $k = 10$, this reduces the cost from $\sim 5 \times 10^{17}$ FLOPs to $\sim 7 \times 10^{12}$ FLOPs—a difference of five orders of magnitude. Libraries such as `scipy.sparse.linalg.svds` and `sklearn.decomposition.TruncatedSVD` provide efficient implementations suitable for the AIether workflow.*

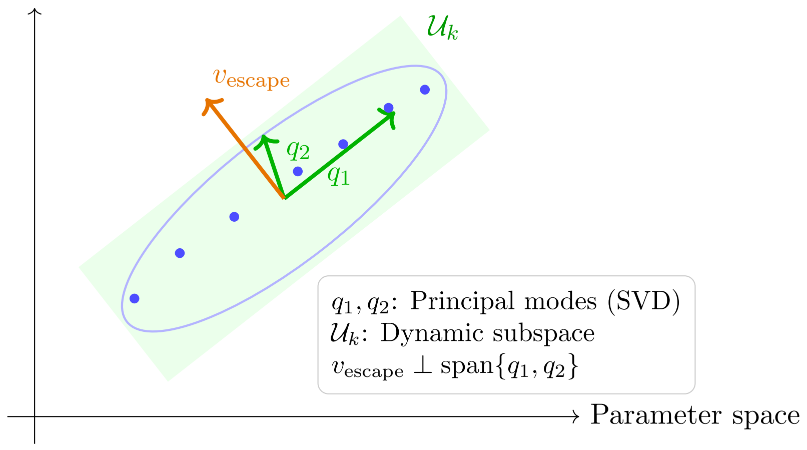
$\mathcal{U}_k$
v_{escape}
q_2
q_1
q_1, q_2: Principal modes (SVD)
$\mathcal{U}_k$: Dynamic subspace
$v_{\text{escape}} \perp \text{span}\{q_1, q_2\}$
Parameter space

Figure 3: Spectral decomposition of the centered velocity field. The trajectory $\{W_0, \ldots, W_{S-1}\}$ (blue points) exhibits structured variance (ellipse) captured by principal modes q_1, q_2 (green arrows). The dynamic subspace $\mathcal{U}_k$ (shaded) spans directions of dominant variation. The escape vector v_{escape} (orange) provides exploration orthogonal to $\mathcal{U}_k$, constructed from secondary spectral modes.

3 Adaptive Geometric Extrapolation

3.1 Foundational Principles

The initialization strategy combines four complementary components: continuity with the last known state (W_{S-1}), incorporation of linear trend through projected mean velocity, curvature correction via acceleration information, and controlled exploration through orthogonal components designed to facilitate escape from potential local minima.

3.2 Projection Framework

Given the dynamic subspace $\mathcal{U}_k$ with projection operator P_k, the trajectory statistics are decomposed into intrinsic and orthogonal components:

$$\bar{V}_{\mathcal{U}} := P_k \operatorname{vec}(\bar{V}), \quad \bar{V}_{\perp} := (I - P_k) \operatorname{vec}(\bar{V}), \tag{3}$$

$$\bar{A}_{\mathcal{U}} := P_k \operatorname{vec}(\bar{A}). \tag{4}$$

For numerical stability and scale control, normalized versions are employed:

$$\hat{V}_{\mathcal{U}} := \frac{\bar{V}_{\mathcal{U}}}{\|\bar{V}_{\mathcal{U}}\| + \varepsilon}, \quad \hat{A}_{\mathcal{U}} := \frac{\bar{A}_{\mathcal{U}}}{\|\bar{A}_{\mathcal{U}}\| + \varepsilon}, \quad \hat{V}_{\perp} := \frac{\bar{V}_{\perp}}{\|\bar{V}_{\perp}\| + \varepsilon}, \tag{5}$$

where $\varepsilon > 0$ is a small regularization constant.

3.3 Spectral Escape Mechanism

When the trajectory remains nearly contained within $\mathcal{U}_k$ (i.e., $\bar{V}_{\perp} \approx 0$), an alternative exploration direction is constructed from secondary spectral modes:

$$v_{\text{escape}} := \frac{\sum_{i=k+1}^{\min(k+\ell,r)} \sigma_i \cdot q_i}{\left\| \sum_{i=k+1}^{\min(k+\ell,r)} \sigma_i \cdot q_i \right\| + \varepsilon},$$

where $q_{k+1}, \ldots, q_{k+\ell}$ denote the subsequent singular vectors and σ_i their associated singular values. This construction leverages directions exhibiting systematic variance while avoiding completely unexplored regions that may induce instability.

3.4 Adaptive Coefficient Design

The extrapolation coefficients dynamically adjust to trajectory characteristics, ensuring robustness across diverse optimization regimes.

Velocity coefficient. The linear trend weight adapts to trajectory efficiency:

$$\beta(\tau) := \frac{\beta_0}{\tau + \varepsilon},$$

where $\beta_0 > 0$ controls the base magnitude. Efficient trajectories ($\tau \approx 1$) yield $\beta \approx \beta_0$, enabling aggressive extrapolation, whereas inefficient trajectories ($\tau \gg 1$) produce $\beta \ll \beta_0$, enforcing conservative steps.

Acceleration coefficient. The curvature correction weight responds to local smoothness:

$$\gamma(\kappa) := \frac{\gamma_0}{1 + \kappa},$$

where $\gamma_0 > 0$ establishes the maximum gain. Smooth trajectories ($\kappa \approx 0$) admit strong second-order corrections ($\gamma \approx \gamma_0$), while highly tortuous paths ($\kappa \gg 1$) attenuate this term ($\gamma \approx 0$) to preserve stability.

Escape coefficient. The exploration term activates selectively under stagnation conditions:

$$\eta(\tau, \kappa) := \begin{cases} 0, & \text{if } \tau < \tau_{\text{crit}} \text{ or } \kappa < \kappa_{\text{crit}}, \\ \eta_0 \cdot (\tau - \tau_{\text{crit}}) \cdot (\kappa - \kappa_{\text{crit}}), & \text{otherwise,} \end{cases}$$

where $\tau_{\text{crit}}, \kappa_{\text{crit}} > 0$ define activation thresholds and $\eta_0 > 0$ scales the response. The multiplicative structure ensures that escape mechanisms engage only when both global inefficiency (high τ) and local tortuosity (high κ) are simultaneously detected, with smooth activation proportional to stagnation severity.

3.5 Unified Extrapolation Formula

Definition 10 (Adaptive Geometric Extrapolation)**.** *Given a trajectory* $\{W_0, \dots, W_{S-1}\} \subset \mathbb{R}^{m \times n}$*, the new layer initialization is defined as:*

$$\boxed{w_{new} = w_{S-1} + \beta(\tau)\hat{V}_{\mathcal{U}} + \gamma(\kappa)\hat{A}_{\mathcal{U}} + \eta(\tau, \kappa)v_{escape}}$$

where $w_{new} := \text{vec}(W_{new})$*, and* $W_{new} := \text{unvec}(w_{new})$ *denotes the initialized weight matrix.*

The formula comprises four distinct components, each serving a specific purpose in the initialization process. The base term w_{S-1} ensures continuity by initializing from the last known state. The linear trend term $\beta\hat{V}_{\mathcal{U}}$ projects the mean velocity onto the dynamic subspace, capturing the dominant direction of parameter evolution. The quadratic correction term $\gamma\hat{A}_{\mathcal{U}}$ anticipates changes in both direction and magnitude of the optimization trajectory. Finally, the exploration term ηv_{escape} introduces an orthogonal component designed to facilitate escape from potential local minima.

3.6 Computational Implementation

The extrapolation procedure requires careful orchestration of several computational steps, each with specific numerical considerations. While the mathematical formulation in the previous section emphasizes geometric intuition, practical implementation must address numerical stability, efficiency, and edge cases that arise with real optimization trajectories.

The algorithm operates in five distinct phases: metric computation quantifies trajectory quality through τ and κ; spectral decomposition identifies the dynamic subspace via truncated SVD; projection and normalization isolate signal from noise while controlling scale; escape direction construction provides exploration when the trajectory is confined; and coefficient adaptation modulates each term's influence based on detected regimes. This sequential structure ensures that each component builds upon validated intermediate results, with regularization applied at critical junctures to prevent numerical degeneracies.

Algorithm 1 details the complete workflow with explicit handling of degenerate cases. Particular attention is given to the escape direction construction (lines 11-17), which adaptively switches between orthogonal complement and secondary spectral modes depending on trajectory confinement. The coefficient computation (lines 19-21) implements the adaptive gains derived in Section 3.4, with threshold-based activation ensuring that aggressive extrapolation engages only under appropriate conditions.

Algorithm 1 Geometric Extrapolation - Tensor-wise Version

```
Require: History $\{W_0, \ldots, W_{S-1}\}$, hyperparameters $\beta_0, \gamma_0, \eta_0, k, \ell$
Ensure: Initialized weights $W_{\text{new}}$
 1: Compute geometric metrics:
 2:     $\tau \leftarrow$ compute_TSF($\{W_t\}$)
 3:     $\kappa \leftarrow$ compute_curvature($\{W_t\}$)
 4: Spectral decomposition:
 5:     $V \leftarrow [V_1, \ldots, V_{S-1}]^\top$                    ▷ Velocity matrix
 6:     $U, \Sigma, Q \leftarrow \text{SVD}(V)$
 7:     $Q_k \leftarrow Q[:, 1:k]$                                    ▷ First $k$ singular vectors
 8: Projection and normalization:
 9:     $\bar{V}_{\mathcal{U}} \leftarrow Q_k Q_k^\top \operatorname{vec}(\bar{V})$
10:     $\bar{A}_{\mathcal{U}} \leftarrow Q_k Q_k^\top \operatorname{vec}(\bar{A})$
11:     $\hat{V}_{\mathcal{U}} \leftarrow \bar{V}_{\mathcal{U}} / (\|\bar{V}_{\mathcal{U}}\| + \varepsilon)$
12:     $\hat{A}_{\mathcal{U}} \leftarrow \bar{A}_{\mathcal{U}} / (\|\bar{A}_{\mathcal{U}}\| + \varepsilon)$
13: Construct escape direction:
14: if $\|(I - P_k)\operatorname{vec}(\bar{V})\| < \varepsilon$ then
15:         $v_{\text{escape}} \leftarrow \sum_{i=k+1}^{k+\ell} \sigma_i q_i$        ▷ Weighted by singular values
16:         $v_{\text{escape}} \leftarrow v_{\text{escape}} / (\|v_{\text{escape}}\| + \varepsilon)$
17: else
18:         $v_{\text{escape}} \leftarrow (I - P_k)\operatorname{vec}(\bar{V})$
19:         $v_{\text{escape}} \leftarrow v_{\text{escape}} / (\|v_{\text{escape}}\| + \varepsilon)$
20: end if
21: Compute adaptive coefficients:
22:     $\beta \leftarrow \beta_0 / (\tau + \varepsilon)$
23:     $\gamma \leftarrow \gamma_0 / (1 + \kappa)$
24:     $\eta \leftarrow \eta_0 \cdot \max(0, \tau - \tau_{\text{crit}}) \cdot \max(0, \kappa - \kappa_{\text{crit}})$
25: Perform extrapolation:
26:     $w_{\text{new}} \leftarrow w_{S-1} + \beta \hat{V}_{\mathcal{U}} + \gamma \hat{A}_{\mathcal{U}} + \eta v_{\text{escape}}$
27: return $W_{\text{new}} = \operatorname{unvec}(w_{\text{new}})$
```

The computational complexity of this procedure is dominated by the SVD operation, which requires $O(SPk)$ operations for a history of S checkpoints, parameter dimension $P = mn$, and subspace dimension k. Modern implementations using randomized SVD or iterative methods (e.g., Lanczos algorithm) can further reduce this cost when $k \ll \min(S, P)$, making the approach practical even for large-scale models.

3.7 Treatment of 1D Tensors (Bias and LayerNorm)

For vector parameters $b \in \mathbb{R}^d$ (bias, normalization scales), the formulation remains valid considering:

1. $\operatorname{vec}(b) = b$ (identity)

2. $\|\cdot\|_F = \|\cdot\|_2$ (Euclidean norm)

3. SVD applied directly on matrix $V \in \mathbb{R}^{(S-1)\times d}$

The adaptation is automatic; the algorithm operates uniformly on 1D and 2D tensors. This universality is a significant advantage of the tensor-wise approach.

4 Architectural Expansion Criteria

4.1 Stagnation Detection

The decision to add a new layer must be based on quantitative evidence of stagnation:

Definition 11 (Stagnation State). *We say that a layer is in **stagnation** at step t if:*

$$\tau_t \geq \tau_{crit} \quad \textit{and} \quad \kappa_t \geq \kappa_{crit},$$

where metrics are computed over a recent sliding window (e.g., last $S = 20$ checkpoints).

Remark 8 (Choice of Thresholds). *Typical values are:*

- $\tau_{crit} \in [1.8, 2.5]$*: Very low values cause premature expansions; very high values ignore genuine stagnations*
- $\kappa_{crit} \in [0.4, 0.8]$*: Balances tolerance for lateral exploration vs. detection of unproductive wandering*

These thresholds can be adjusted empirically for each application domain.

Definition 12 (Persistent Stagnation Counter). *We maintain a counter c_{stag} that increments at each evaluation where the stagnation criterion is satisfied, and resets to 0 otherwise.*

*The condition to **trigger growth** is:*

$$c_{stag} \geq p_{patience},$$

where $p_{patience} \in \{3, 5, 10\}$ controls tolerance before architectural intervention.

Remark 9 (Counter Justification). *The counter avoids reactions to temporary fluctuations. Requiring persistent evidence ($p_{patience}$ consecutive evaluations) ensures that stagnation is genuine and not an artifact of training noise or temporary exploratory phases of the optimizer.*

4.2 Growth Protocol

Algorithm 2 Procedural Growth Protocol

```
Require: Current model with $L$ layers, frontier layer $\ell_{\text{front}}$
Ensure: Expanded model with new layer $\ell_{\text{new}} = \ell_{\text{front}} - 1$
 1: Phase 1: Freezing and Checkpoint
 2:     Save state $L_1$ of layer $\ell_{\text{front}}$
 3:     Freeze all active layers: $\forall \ell \in \mathcal{L}_{\text{active}} : \nabla_{\theta_\ell} \mathcal{L} \leftarrow 0$
 4: Phase 2: Geometric Initialization
 5:     Retrieve history $\mathcal{H}_{\ell_{\text{front}}} = \{W_0, \ldots, W_{S-1}\}$
 6:     $W_{\text{new}} \leftarrow \text{GeometricExtrapolator}(\mathcal{H}_{\ell_{\text{front}}})$
 7:     Initialize layer $\ell_{\text{new}}$ with $W_{\text{new}}$
 8: Phase 3: Isolated Warmup
 9: for $t = 1$ to $N_{\text{warmup}}$ do                    ▷ Typically $N_{\text{warmup}} = 500$
10:     Train only $\theta_{\ell_{\text{new}}}$ with reduced LR ($\sim 0.1\times$ normal LR)
11: end for
12: Phase 4: Unfreezing and Resumption
13:     Unfreeze all layers: $\forall \ell \in \mathcal{L}_{\text{active}} \cup \{\ell_{\text{new}}\}$
14:     Restart LR scheduler with warmup
15:     Resume full training
16: return Expanded model
```

The training procedure consists of four distinct phases, each serving a specific purpose in the integration of new layers. **Phase 1** employs freezing to prevent perturbation of already-trained layers during new layer integration. **Phase 2** utilizes geometric extrapolation for informed initialization of the newly added layer. **Phase 3** implements an isolated warmup period, allowing the new layer to adapt without competing with updates to other layers. Finally, **Phase 4** introduces gradual unfreezing to resume joint training, enabling co-adaptation of all layers in the expanded architecture.

5 Analysis and Justifications

5.1 Continuity Preservation

A critical objective of the proposed method is to minimize functional perturbation when adding a layer. Let $f_{\text{old}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ denote the function represented before expansion, and f_{new} denote the function after adding the new layer. Geometric extrapolation aims to ensure that

$$\mathbb{E}_{x \sim \mathcal{D}} \left[\| f_{\text{new}}(x) - f_{\text{old}}(x) \|^2 \right] \text{ remains small.}$$

This property is achieved through three mechanisms:

1. The initialization begins from W_{S-1}, which already represents a trained configuration;
2. The additional terms ($\beta \hat{V}_{\mathcal{U}}$ and $\gamma \hat{A}_{\mathcal{U}}$) constitute projections along directions of historical variation, thereby providing stability;

3. Normalization constrains the magnitude of perturbation.

In contrast, random initialization may induce large and unpredictable functional perturbations.

5.2 Convergence Efficiency

The central hypothesis underlying this approach is that initialization within a favorable region of parameter space reduces the number of iterations required for convergence.

Several factors contribute to this acceleration:

- **Prior information**: Extrapolation incorporates knowledge about productive directions from the optimization trajectory;
- **Plateau avoidance**: The escape term facilitates navigation away from low-gradient regions;
- **Continuity**: Reduced time is required for recalibration following layer addition.

The magnitude of acceleration depends on:

- The quality of the historical trajectory, quantified by signal-to-noise ratio $\text{SNR} := \|\bar{V}_{\mathcal{U}}\|/\|\bar{V}_{\perp}\|$;
- The smoothness of the loss landscape in the neighborhood of initialization;
- The appropriateness of hyperparameters β_0, γ_0, and η_0.

5.3 Limitations and Failure Modes

Several scenarios may limit the effectiveness of geometric extrapolation:

1. **Highly non-convex landscapes**: When the loss surface exhibits numerous qualitatively different local minima, extrapolation may direct optimization toward suboptimal regions;
2. **Regime shifts**: If optimization dynamics undergo substantial changes after expansion (e.g., due to data distribution shifts), the historical trajectory may lack predictive value;
3. **Suboptimal subspace dimensionality**: An excessively small subspace (low k) may discard critical information, while an excessively large subspace may incorporate noise.

In such cases, more conservative initialization strategies (e.g., layer copying or standard initialization with small perturbations) may prove more effective.

6 Practical Considerations

6.1 Hyperparameter Configuration

The AIether framework introduces several hyperparameters that control extrapolation behavior, stagnation detection, and architectural growth dynamics. The recommended values in Table 1 were established through empirical tuning on transformer language models (10M–50M parameters) trained on FineWebEdu subsets [15], with checkpoints collected every 50–100 optimization steps.

Parameter	**Range**	**Default Value**
β_0	$[0.05, 0.5]$	0.1
γ_0	$[0.05, 0.3]$	0.15
η_0	$[0.01, 0.1]$	0.03
k (subspace dim.)	$[5, 30]$	10
ℓ (escape modes)	$[3, 10]$	5
τ_{crit}	$[1.5, 3.0]$	2.0
κ_{crit}	$[0.3, 1.0]$	0.5
p_{patience}	$[3, 10]$	5
S (history window)	$[5, 15]$	5
N_{warmup}	$[300, 800]$	500
ε (regularization)	10^{-6} – 10^{-8}	10^{-8}

Table 1: AIether hyperparameters with recommended ranges established on 10M–50M parameter language models.

6.1.1 Extrapolation Coefficients

The base gains β_0, γ_0, and η_0 control the magnitude of linear, curvature, and escape components respectively. The default $\beta_0 = 0.1$ produces conservative extrapolation aligned with initialization guidelines [10, 12], which suggest maintaining parameter perturbations within $O(0.1)$ standard deviations to preserve gradient flow. The acceleration gain $\gamma_0 = 0.15$ is deliberately smaller than β_0 to prevent over-correction in high-curvature regions where discrete sampling artifacts degrade acceleration estimates. The escape gain $\eta_0 = 0.03$ generates modest orthogonal perturbations sufficient for saddle point escape [13] without inducing parameter drift.

6.1.2 Subspace Dimensions

The dynamic subspace dimension $k = 10$ and escape mode count $\ell = 5$ reflect empirical observations that neural network optimization explores low-dimensional subspaces even in high-dimensional parameter spaces [14, 5]. Studies consistently find effective dimensionalities of $O(10)$–$O(100)$ during gradient descent. The choice $k = 10$ captures dominant trajectory modes while remaining computationally efficient, with secondary modes ($\ell = 5$) providing structured exploration beyond the primary subspace.

6.1.3 Stagnation Detection

The thresholds $\tau_{\text{crit}} = 2.0$ and $\kappa_{\text{crit}} = 0.5$ balance sensitivity to genuine plateaus against tolerance for transient exploration. Empirical observations during preliminary experiments revealed: $\tau \in [1.0, 1.5]$ during efficient convergence, $\tau \in [1.5, 2.5]$ during stable training with healthy exploration, and $\tau > 3.0$ preceding training plateaus. Similarly, $\kappa_{\text{crit}} = 0.5$ detects lateral wandering where perpendicular movement constitutes half of parallel progress—distinguishing productive stochastic exploration from pathological oscillation.

The patience requirement $p_{\text{patience}} = 5$ consecutive evaluations prevents false positives from temporary fluctuations. With typical evaluation intervals of 500–1000 steps, this corresponds to 2,500–5,000 steps of sustained stagnation before triggering expansion.

6.1.4 History Window and Warmup

The history window $S = 5$ checkpoints balances temporal resolution for velocity/acceleration estimation against memory efficiency and adaptation speed. Longer windows ($S > 10$) improve statistical robustness but risk incorporating outdated dynamics from earlier training regimes, particularly during learning rate schedules or curriculum learning.

The warmup duration $N_{\text{warmup}} = 500$ steps with reduced learning rate (typically $0.1\times$ nominal) allows new layers to calibrate without destabilizing the existing model. This aligns with progressive training methodologies [9, 1] that advocate gradual capacity integration. Preliminary experiments showed warmup periods below 300 steps frequently produced gradient spikes, while durations exceeding 800 steps provided diminishing returns.

6.1.5 Domain Adaptation

While default hyperparameters prove robust within the experimental regime, substantial architectural or domain changes may require adjustment. Models exceeding 1B parameters may benefit from larger k to capture richer trajectory structure, while domains with sparse gradients (e.g., reinforcement learning) may require more conservative β_0 and higher τ_{crit} thresholds to avoid premature expansion during exploratory phases.

6.2 Computational Cost

Proposition 3 (Extrapolation Complexity). *For a layer with $P = m \times n$ parameters and history size S:*

- ***Geometric metrics*** *(τ, κ): $O(SP)$*
- ***Truncated SVD*** *(first k vectors): $O(SP \cdot k)$ using iterative algorithms*
- ***Projections****: $O(Pk)$*
- ***Total per layer****: $O(SP \cdot k)$*

For models with L layers and sequential growth, the accumulated cost is $O(L \cdot SP \cdot k)$, negligible compared to training cost ($O(N_{iter} \cdot L \cdot P \cdot B)$ for batch size B and N_{iter} iterations).

6.3 Memory Management

Remark 10 (History Storage)**.** *Complete history of S checkpoints per layer requires $O(SP)$ memory. For large models (billions of parameters), compression strategies are necessary:*

1. ***Quantization**: Store checkpoints in FP16 or INT8 (2-4x reduction)*
2. ***Adaptive sampling**: Collect checkpoints with decreasing frequency:*
   - *Recent: every 50 steps*
   - *Medium: every 200 steps*
   - *Old: every 1000 steps*
3. ***Disk streaming**: Keep history in persistent storage (SSD), load on demand for metric computation*
4. ***Rolling history**: Maintain only the S most recent checkpoints, discarding older ones*

For a 7B parameter model with $S = 20$ and FP16:

$$\textit{Memory} = 7 \times 10^9 \times 2 \ \textit{bytes} \times 20 = 280 \ \textit{GB}$$

This is manageable with disk streaming or additional compression.

7 Future Directions and Open Investigations

AIether has established a solid foundation for geometric initialization in sequential depth. However, large-scale practical application raises critical questions regarding optimization dynamics and alternative topologies that warrant detailed investigation.

7.1 Learning Rate and Momentum Dynamics

A central hypothesis of AIether is that an initialization aligned with the historical trajectory reduces the need for conservative warmup phases.

- **Momentum Injection**: Investigate whether geometric initialization allows maintaining (or even increasing) the Learning Rate (η) immediately after expansion, leveraging the accumulated momentum (μ) from previous layers, in contrast to the standard practice of "resetting" the optimizer for the new layer.
- **Resilience to Aggressive LRs**: Empirically quantify whether models expanded via AIether support learning rates $10\times$ larger without numerical divergence, accelerating the transition through saddle regions.

7.2 Retroactive Coupling (L vs $L + 1$)

The insertion of layer $L + 1$ alters the loss landscape perceived by layer L. Geometric extrapolation must mitigate the gradient shock.

- **Gradient Conditioning**: Analyze how the spectral structure of the new layer affects the Hessian condition number in lower layers. The goal is to ensure that initialization does not introduce destructive noise that degrades what layer L has already learned (the problem of "retroactive catastrophic interference").
- **Subspace Alignment**: Verify whether the principal activation eigenvectors of layer L remain aligned with those of layer $L + 1$ post-initialization, preserving signal flow.

7.3 Metric Stability and Calibration

For continuous operation (Lifelong Learning), metrics τ and κ must be robust to regime shifts.

- **Temporal Scale Invariance**: Determine whether critical thresholds ($\tau_{\text{crit}}, \kappa_{\text{crit}}$) require dynamic readjustment as network depth increases or if a universal normalization exists that renders them constant throughout training.
- **False Positive Detection**: Refine the calculation of κ to distinguish between "healthy exploration" (stochastic SGD noise) and "pathological oscillation" (actual stagnation), avoiding unnecessary expansions on temporary plateaus.

7.4 Non-Sequential Growth Topologies

Although this work focuses on depth (vertical growth), the logic of geometric extrapolation is agnostic to tensor direction.

- **Horizontal Expansion (Width)**: Apply spectral extrapolation to add neurons or attention heads laterally, allowing the model to specialize sub-networks for different data domains without increasing the critical path depth.
- **Diagonal Growth**: Investigate architectures that grow simultaneously in depth and width, or that add dense residual connections (DenseNet-style) based on the detection of information bottlenecks in the optimization trajectory.

8 Conclusion

We present **AIether**, a framework for the procedural growth of neural networks via adaptive geometric extrapolation. The method is based on fundamental principles:

- **Geometric analysis**: Quantitative characterization of the optimization trajectory via metrics τ, κ, and ρ.

- **Spectral decomposition**: Identification of dynamic subspaces via SVD to capture learning patterns.
- **Adaptive extrapolation**: Informed initialization combining linear trend, curvature correction, and subspace exploration.
- **Stagnation detection**: Quantitative and persistent criteria to trigger architectural interventions.

AIether targets three main objectives:

1. **Accelerate convergence** post-expansion through initialization informed by historical dynamics.
2. **Preserve functional continuity** to minimize destructive perturbation to the existing model.
3. **Detect and escape** stagnation regions in parameter space.

The approach is characterized by being:

- **Tensor-wise**: Operates individually on each tensor, respecting specific geometric structures.
- **Efficient**: Computational complexity of $O(SPk)$ per layer, negligible compared to training costs.
- **Scalable**: Applicable to large-scale models through appropriate compression strategies.
- **Modular**: Can be integrated into existing training pipelines without requiring deep architectural modifications.

While extensive experimental validation is necessary to cover all domains, the mathematical foundation and detailed practical considerations establish a basis for future investigations. Implementation code and experiments will be made available in a public repository to ensure reproducibility and enable community extension.

Acknowledgments

This work was developed as part of the AIether research project on geometric optimization of deep neural networks. We thank the deep learning community for tools and frameworks that make this research possible.

References

[1] Rusu, A. A., et al. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671.*

[2] Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3), 211–218.

[3] Chen, T., Goodfellow, I., & Shlens, J. (2015). Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641.*

[4] Benhamou, S. (2004). How to reliably estimate the tortuosity of an animal's path: straightness, sinuosity, or fractal dimension? *Journal of Theoretical Biology*, 229(2), 209–220.

[5] Gur-Ari, G., Roberts, D. A., & Dyer, E. (2018). Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754.*

[6] Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2), 217–288.

[7] do Carmo, M. P. (1992). *Riemannian Geometry.* Birkhäuser Boston.

[8] Bronstein, M. M., et al. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478.*

[9] Karras, T., et al. (2017). Progressive growing of GANs for improved quality, stability, and variation. *ICLR 2018.*

[10] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *AISTATS 2010.*

[11] Saxe, A. M., et al. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120.*

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

[13] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 27.

[14] Li, C., Farkhoor, H., Liu, R., & Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. *International Conference on Learning Representations (ICLR).*

[15] Penedo, G., et al. (2024). The FineWeb Datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557.*

[16] Fort, S., & Ganguli, S. (2019). Emergent properties of the local geometry of neural loss landscapes. *arXiv preprint arXiv:1910.05929.*

[17] Raghu, M., et al. (2017). SVCCA: Singular Vector Canonical Correlation Analysis for deep learning dynamics and interpretability. *NeurIPS 2017.*

A Complete System Pseudocode

Algorithm 3 AIether - Complete Procedural Growth System

```
Require: Initial model $M_0$, dataset $\mathcal{D}$, hyperparameters $\mathcal{H}$
Ensure: Trained model $M_{\text{final}}$
 1: $M \leftarrow M_0$
 2: $\mathcal{H}_{\text{layer}} \leftarrow \{\}$                          ▷ Histories per layer
 3: $c_{\text{stag}} \leftarrow 0$
 4: while not converged do
 5:     for $t = 1$ to $N_{\text{steps\_per\_eval}}$ do
 6:         Train $M$ for 1 step on $\mathcal{D}$
 7:         if $t \bmod \Delta_{\text{collect}} = 0$ then
 8:             Collect snapshot of frontier layer $\ell_{\text{front}}$
 9:             $\mathcal{H}_{\text{layer}}[\ell_{\text{front}}]$.append($W_{\ell_{\text{front}}}$)
10:         end if
11:     end for
12:     Evaluation and Stagnation Check:
13:     $\tau \leftarrow$ compute_TSF($\mathcal{H}_{\text{layer}}[\ell_{\text{front}}]$)
14:     $\kappa \leftarrow$ compute_curvature($\mathcal{H}_{\text{layer}}[\ell_{\text{front}}]$)
15:     if $\tau \geq \tau_{\text{crit}}$ and $\kappa \geq \kappa_{\text{crit}}$ then
16:         $c_{\text{stag}} \leftarrow c_{\text{stag}} + 1$
17:     else
18:         $c_{\text{stag}} \leftarrow 0$
19:     end if
20:     if $c_{\text{stag}} \geq p_{\text{patience}}$ then
21:         Trigger Growth:
22:         $W_{\text{new}} \leftarrow$ GeometricExtrapolator($\mathcal{H}_{\text{layer}}[\ell_{\text{front}}], \mathcal{H}$)
23:         $M \leftarrow$ GrowModel($M, W_{\text{new}}$)
24:         $c_{\text{stag}} \leftarrow 0$
25:         $\mathcal{H}_{\text{layer}}[\ell_{\text{front}} - 1] \leftarrow [\,]$          ▷ Initialize new history
26:     end if
27: end while
28: return $M$
```

B Implementation Details

B.1 TSF Computation Function

Algorithm 4 Compute TSF (Temporal Stretch Factor)

```
Require: History $\{W_0, \dots, W_{S-1}\}$, $\varepsilon = 10^{-8}$
Ensure: $\tau \in [1, \infty)$
1: $\Delta \leftarrow W_{S-1} - W_0$
2: $\mathcal{L}_{\text{direct}} \leftarrow \|\Delta\|_F$
3: $\mathcal{L}_{\text{path}} \leftarrow 0$
4: for $t = 1$ to $S - 1$ do
5:     $V_t \leftarrow W_t - W_{t-1}$
6:     $\mathcal{L}_{\text{path}} \leftarrow \mathcal{L}_{\text{path}} + \|V_t\|_F$
7: end for
8: $\tau \leftarrow \mathcal{L}_{\text{path}} / (\mathcal{L}_{\text{direct}} + \varepsilon)$
9: return $\tau$
```

B.2 Curvature Computation Function

Algorithm 5 Compute Curvature (Effective Curvature)

```
Require: History $\{W_0, \dots, W_{S-1}\}$, $\varepsilon = 10^{-8}$
Ensure: $\kappa \in [0, \infty)$
1: $\Delta \leftarrow W_{S-1} - W_0$
2: $\hat{\Delta} \leftarrow \Delta / (\|\Delta\|_F + \varepsilon)$
3: sum_parallel $\leftarrow 0$
4: sum_perp $\leftarrow 0$
5: for $t = 1$ to $S - 1$ do
6:     $V_t \leftarrow W_t - W_{t-1}$
7:     $V_t^{\|} \leftarrow \langle V_t, \hat{\Delta} \rangle_F \cdot \hat{\Delta}$
8:     $V_t^{\perp} \leftarrow V_t - V_t^{\|}$
9:     sum_parallel $\leftarrow$ sum_parallel $+ \|V_t^{\|}\|_F$
10:    sum_perp $\leftarrow$ sum_perp $+ \|V_t^{\perp}\|_F$
11: end for
12: $\kappa \leftarrow$ sum_perp/(sum_parallel $+ \varepsilon$)
13: return $\kappa$
```