AIether: Procedural Growth of Neural Networks via Adaptive Geometric Extrapolation

Samuel Rocha

December 2025

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

2.2.1 Discrete Velocity and Acceleration

To quantify dynamic behavior, we define discrete approximations of temporal derivatives:

Definition 3 (Discrete Velocity)**.** *The velocity between consecutive steps is:*

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

The mean velocity along the trajectory is:

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

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

Definition 4 (Discrete Acceleration)**.** *The acceleration (variation of velocity) is:*

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

The mean acceleration is:

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

Remark 2. *These definitions are discrete analogs of first and second derivatives. Velocity captures the rate of change of parameters; acceleration captures changes in the direction or magnitude of this rate. High acceleration indicates that the optimizer is constantly changing direction, suggesting difficulty in finding a consistent productive direction.*

2.2.2 Temporal Stretch Factor (TSF)

A crucial global metric is trajectory efficiency:

Definition 5 (Path Length and TSF)**.** *The total 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 regularization term (typically $\varepsilon = 10^{-8}$).

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

1. ***Lower bound**: $\tau \geq 1$ always, with equality if and only if the trajectory is a line segment.*
2. ***Reparametrization invariance**: τ is invariant under uniform time rescaling.*
3. ***Physical interpretation**: $\tau - 1$ measures the fraction of "wasted movement" (oscillations, backtracking) relative to useful displacement.*

Remark 3 (Practical Interpretation)**.**

- $\tau \approx 1$*: Efficient trajectory, nearly rectilinear movement*
- $\tau \in [1.5, 2.5]$*: Moderate tortuosity, typical of healthy optimization*
- $\tau > 3$*: Strong evidence of stagnation or excessive oscillation*

2.2.3 Effective Curvature

Beyond global efficiency, we need to quantify local tortuosity:

Definition 6 (Parallel-Perpendicular Decomposition)**.** *For each increment V_t, we decompose relative to the global 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 $\langle A, B \rangle_F := \mathrm{Tr}(A^\top B)$ is the Frobenius inner product.

Definition 7 (Effective Curvature)**.** *The* ***effective curvature*** *measures the ratio between perpendicular movement (lateral deviation) and parallel movement (useful progress):*

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

Remark 4 (Interpretation of κ)**.**

- $\kappa \approx 0$*: Nearly rectilinear trajectory in the* Δ *direction*
- $\kappa \in [0.3, 0.8]$*: Moderate curvature, healthy lateral exploration*
- $\kappa > 1$*: Movement predominantly orthogonal to net progress, possible stagnation*

The curvature is dimensionless and invariant under global rescaling of W*. High values indicate that the optimizer is "wandering" laterally without making consistent progress in the direction of net improvement.*

2.3 Spectral Analysis of the Trajectory

2.3.1 Temporal Covariance Matrix

To understand the structure of parameter variation over time:

Definition 8 (Temporal Covariance)**.** *Let* $\bar{W} := \frac{1}{S}\sum_{t=0}^{S-1} W_t$ *be the temporal mean. For matrices* $W_t \in \mathbb{R}^{m\times n}$*, we vectorize* $w_t := \mathrm{vec}(W_t) \in \mathbb{R}^{mn}$ *and define:*

$$\Sigma_{real} := \frac{1}{S}\sum_{t=0}^{S-1}(w_t - \bar{w})(w_t - \bar{w})^{\top} \in \mathbb{R}^{mn\times mn},$$

where $\bar{w} := \mathrm{vec}(\bar{W})$*.*

To establish a baseline, we consider the covariance of an ideal trajectory:

Definition 9 (Ideal Covariance (Baseline))**.** *For a perfectly linear trajectory between* W_0 *and* W_{S-1}*, the ideal covariance is:*

$$\Sigma_{ideal} := \frac{1}{4}\cdot \mathrm{vec}(\Delta)\,\mathrm{vec}(\Delta)^{\top},$$

which is a rank-1 matrix corresponding to the variance of a uniform distribution over the endpoints.

Remark 5 (Fundamental Variance Property)**.** *For any trajectory with fixed endpoints, we have:*

$$\mathrm{Tr}(\Sigma_{real}) \geq \mathrm{Tr}(\Sigma_{ideal}) = \frac{1}{4}\|\Delta\|_F^2$$

This inequality follows from the decomposition into parallel and perpendicular components: the real variance always includes additional variance from lateral deviations.

Definition 10 (Spectral Tortuosity Factor)**.**

$$\boxed{\rho := \frac{\mathrm{Tr}(\Sigma_{real})}{\mathrm{Tr}(\Sigma_{ideal})}}$$

measures the excess accumulated variance relative to the theoretical minimum.

Remark 6 (Relationship between ρ, τ, and κ)**.** *The three metrics capture complementary aspects:*

- τ*: global inefficiency (path length vs. direct distance)*
- κ*: local tortuosity (accumulated perpendicular deviation)*
- ρ*: spectral dispersion (accumulated variance vs. minimum variance)*

In general: $\rho \geq 1$ *and* $\tau \geq 1$*, with high values indicating complex/stagnant dynamics. We typically observe positive correlation between these metrics, but each can reveal distinct aspects of optimization behavior.*

2.3.2 Decomposition into Dynamic Subspaces

Spectral analysis via SVD identifies the principal directions of variation:

Definition 11 (Dynamic Subspace via SVD)**.** *Let* $V \in \mathbb{R}^{(S-1)\times mn}$ *be the matrix whose rows are* $\mathrm{vec}(V_t)^\top$*. The singular value decomposition:*

$$V = U\Sigma Q^\top,$$

where $Q = [q_1, \ldots, q_r] \in \mathbb{R}^{mn\times r}$ *contains the right singular vectors, ordered by decreasing singular values* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$*.*

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

$$\mathcal{U}_k := \mathrm{span}\{q_1, \ldots, q_k\},$$

and the corresponding orthogonal projector is:

$$P_k := Q_k Q_k^\top, \quad \textit{where} \quad Q_k := [q_1, \ldots, q_k].$$

Remark 7 (Interpretation of Dynamic Subspace)**.** *The vectors* $q_1, \ldots, q_k$ *represent the* k *directions of greatest variance in the velocity trajectory. These directions capture:*

- *Systematic patterns of parameter change*
- *Dominant modes of optimizer exploration*
- *Low-dimensional structure in learning dynamics*

The choice of k *is a hyperparameter that controls the trade-off between fidelity to historical trajectory and capacity for generalization/exploration.*

Remark 8 (Optimality of SVD Projection)**.** *Among all k-dimensional subspaces, $\mathcal{U}_k$ minimizes the reconstruction error:*

$$\mathcal{U}_k = \operatorname*{argmin}_{\dim(\mathcal{U})=k} \sum_{t=1}^{S-1} \| \operatorname{vec}(V_t) - P_{\mathcal{U}} \operatorname{vec}(V_t)\|^2,$$

where $P_{\mathcal{U}}$ is the orthogonal projector onto $\mathcal{U}$. This is a consequence of the Eckart-Young-Mirsky theorem.

3 Adaptive Geometric Extrapolation

3.1 Multi-Scale Extrapolation Principle

The core idea is to construct an initialization that combines:

1. **Continuity**: Start from the last known state (W_{S-1})
2. **Linear trend**: Add projected mean velocity component
3. **Curvature correction**: Incorporate acceleration information to anticipate changes
4. **Exploration**: Include orthogonal component to escape possible local minima

Definition 12 (Projection Operators)**.** *Given the dynamic subspace $\mathcal{U}_k$, we define:*

$$\bar{V}_{\mathcal{U}} := P_k \operatorname{vec}(\bar{V}), \quad \textit{(projected velocity)} \tag{3}$$

$$\bar{A}_{\mathcal{U}} := P_k \operatorname{vec}(\bar{A}), \quad \textit{(projected acceleration)} \tag{4}$$

$$\bar{V}_{\perp} := (I - P_k) \operatorname{vec}(\bar{V}). \quad \textit{(orthogonal component)} \tag{5}$$

Definition 13 (Normalized Versions)**.** *For scale control and numerical stability:*

$$\hat{V}_{\mathcal{U}} := \frac{\bar{V}_{\mathcal{U}}}{\|\bar{V}_{\mathcal{U}}\| + \varepsilon}, \tag{6}$$

$$\hat{A}_{\mathcal{U}} := \frac{\bar{A}_{\mathcal{U}}}{\|\bar{A}_{\mathcal{U}}\| + \varepsilon}, \tag{7}$$

$$\hat{V}_{\perp} := \frac{\bar{V}_{\perp}}{\|\bar{V}_{\perp}\| + \varepsilon}. \tag{8}$$

3.2 Spectral Escape Direction

In situations where the orthogonal component is very small (trajectory nearly contained in $\mathcal{U}_k$), we need an alternative exploration mechanism:

Definition 14 (Escape via Secondary Modes)**.** *To avoid degeneracy when $\bar{V}_{\perp} \approx 0$, we construct an escape direction using the next ℓ singular vectors:*

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

where the weights σ_i emphasize directions of greater secondary variance.

Remark 9 (Geometric Justification)**.** *The vectors $q_{k+1}, \ldots, q_{k+\ell}$ represent directions of variance "filtered" by projection onto $\mathcal{U}_k$, but which still contain information about exploratory modes of the optimizer. This construction is superior to random perturbations because it:*

- *Respects the spectral structure of the historical trajectory*
- *Prioritizes directions that have already shown some systematic variance*
- *Avoids completely unexplored regions that may be unstable*

3.3 Adaptive Coefficients

Extrapolation coefficients must adapt to trajectory characteristics:

Definition 15 (Linear Gain β)**.** *The coefficient of the velocity term adapts to trajectory efficiency:*

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

where $\beta_0 > 0$ is the base gain.

***Rationale**:*

- $\tau \approx 1$ *(efficient trajectory)* $\Rightarrow \beta \approx \beta_0$ *(larger extrapolated step)*
- $\tau \gg 1$ *(inefficient trajectory)* $\Rightarrow \beta \ll \beta_0$ *(conservative extrapolation)*

Efficient trajectories merit confidence for aggressive extrapolation; tortuous trajectories require caution.

Definition 16 (Curvature Gain γ)**.** *The coefficient of the acceleration term adapts to local smoothness:*

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

where $\gamma_0 > 0$ is the maximum gain.

***Rationale**:*

- $\kappa \approx 0$ *(smooth trajectory)* $\Rightarrow \gamma \approx \gamma_0$ *(aggressive second-order correction)*
- $\kappa \gg 1$ *(tortuous trajectory)* $\Rightarrow \gamma \approx 0$ *(attenuation to avoid instability)*

Acceleration information is reliable only when the trajectory is locally smooth.

Definition 17 (Escape Gain η)**.** *The coefficient of the orthogonal term activates in stagnation regime:*

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

where $\tau_{crit}, \kappa_{crit} > 0$ are activation thresholds.

***Rationale**: Orthogonal escape is an aggressive intervention that should only be activated when there is simultaneous evidence of:*

- *Global inefficiency (high τ)*
- *Local tortuosity (high κ)*

The product $(\tau - \tau_{crit}) \cdot (\kappa - \kappa_{crit})$ ensures smooth activation proportional to stagnation severity.

3.4 Unified Extrapolation Formula

Definition 18 (Adaptive Geometric Extrapolation)**.** *Given a trajectory* $\{W_0, \ldots, W_{S-1}\} \subset \mathbb{R}^{m\times n}$*, the new layer initialization is:*

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

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

Remark 10 (Term Decomposition)**.**

- ***Term 0*** *(w_{S-1}): Continuity - start from last known state*
- ***Term 1*** *($\beta\hat{V}_{\mathcal{U}}$): Linear trend - project mean velocity onto dynamic subspace*
- ***Term 2*** *($\gamma\hat{A}_{\mathcal{U}}$): Quadratic correction - anticipate changes in direction/magnitude*
- ***Term 3*** *(ηv_{escape}): Exploration - add orthogonal component for local minimum escape*

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_{new}

Compute geometric metrics:
  $\tau \leftarrow \text{compute\_TSF}(\{W_t\})$
  $\kappa \leftarrow \text{compute\_curvature}(\{W_t\})$
Spectral decomposition:
  $V \leftarrow [V_1, \ldots, V_{S-1}]^\top$ ▷ Velocity matrix
  $U, \Sigma, Q \leftarrow \mathrm{SVD}(V)$
  $Q_k \leftarrow Q[:, 1:k]$ ▷ First k singular vectors
Projection and normalization:
  $\bar{V}_{\mathcal{U}} \leftarrow Q_k Q_k^\top \mathrm{vec}(\bar{V})$
  $\bar{A}_{\mathcal{U}} \leftarrow Q_k Q_k^\top \mathrm{vec}(\bar{A})$
  $\hat{V}_{\mathcal{U}} \leftarrow \bar{V}_{\mathcal{U}}/(\|\bar{V}_{\mathcal{U}}\| + \varepsilon)$
  $\hat{A}_{\mathcal{U}} \leftarrow \bar{A}_{\mathcal{U}}/(\|\bar{A}_{\mathcal{U}}\| + \varepsilon)$
Construct escape direction:
if $\|(I - P_k)\,\mathrm{vec}(\bar{V})\| < \varepsilon$ **then**
    $v_{\text{escape}} \leftarrow \sum_{i=k+1}^{k+\ell} \sigma_i q_i$ ▷ Normalized
else
    $v_{\text{escape}} \leftarrow (I - P_k)\,\mathrm{vec}(\bar{V})$ ▷ Normalized
end if
Compute coefficients:
  $\beta \leftarrow \beta_0/(\tau + \varepsilon)$
  $\gamma \leftarrow \gamma_0/(1 + \kappa)$
  $\eta \leftarrow \eta_0 \cdot \max(0, \tau - \tau_{\text{crit}}) \cdot \max(0, \kappa - \kappa_{\text{crit}})$
Extrapolation:
  $w_{\text{new}} \leftarrow w_{S-1} + \beta\hat{V}_{\mathcal{U}} + \gamma\hat{A}_{\mathcal{U}} + \eta v_{\text{escape}}$
return $W_{\text{new}} = \mathrm{unvec}(w_{\text{new}})$

3.5 Treatment of 1D Tensors (Bias and LayerNorm)

Remark 11 (Vector Case)**.** *For vector parameters $b \in \mathbb{R}^d$ (bias, normalization scales), the formulation remains valid considering:*

- $\text{vec}(b) = b$ *(identity)*
- $\|\cdot\|_F = \|\cdot\|_2$ *(Euclidean norm)*
- *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 19 (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 12 (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 20 (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 13 (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 ℓ_{front}
Ensure: Expanded model with new layer $\ell_{\text{new}} = \ell_{\text{front}} - 1$

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

Remark 14 (Protocol Details)**.**

- ***Phase 1****: Freezing prevents perturbation of already-trained layers during new layer integration*
- ***Phase 2****: Uses geometric extrapolation for informed initialization*
- ***Phase 3****: Isolated warmup allows the new layer to adapt without competing with updates to other layers*
- ***Phase 4****: Gradual unfreezing resumes joint training with co-adaptation of all layers*

5 Analysis and Justifications

5.1 Continuity Preservation

Remark 15 (Approximate Functional Continuity)**.** *A critical objective is to minimize functional perturbation when adding a layer. Let $f_{old} : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ be the function represented before expansion, and f_{new} after adding the new layer.*

Geometric extrapolation aims to ensure that:

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

Intuitively, this works because:

1. *We start from W_{S-1}, which already represents a trained configuration*

2. *The additional terms ($\beta\hat{V}_{\mathcal{U}}$, $\gamma\hat{A}_{\mathcal{U}}$) are projections in directions of historical variation, hence "safe"*

3. *Normalization limits the magnitude of perturbation*

In contrast, random initialization can cause large and unpredictable functional perturbations.

5.2 Convergence Efficiency

Remark 16 (Convergence Acceleration)**.** *The central hypothesis is that initializing in a "good" region of parameter space reduces the number of iterations required for convergence.*

Factors contributing to acceleration:

- ***Prior information**: Extrapolation incorporates knowledge about productive directions*
- ***Plateau avoidance**: The escape term helps avoid low-gradient regions*
- ***Continuity**: Less time spent on "recalibration" after adding the layer*

The magnitude of acceleration depends on:

- *Quality of historical trajectory (high $SNR := \|\bar{V}_{\mathcal{U}}\|/\|\bar{V}_{\perp}\|$)*
- *Smoothness of the loss landscape in the neighborhood of initialization*
- *Adequacy of hyperparameters $\beta_0, \gamma_0, \eta_0$*

5.3 Limitations and Failure Cases

Remark 17 (Problematic Scenarios)**.**

1. ***Highly non-convex landscapes**: If the loss surface has many radically different local minima, extrapolation may lead to poor regions*
2. ***Regime shifts**: If optimization dynamics change drastically after expansion (e.g., due to data distribution change), historical trajectory may not be informative*
3. ***Short or noisy history**: With few checkpoints ($S < 10$) or very noisy trajectory, geometric metrics have high variance*
4. ***Inadequate choice of** k: Too small subspace (low k) may lose critical information; too large may include noise*

In such cases, more conservative initialization (e.g., layer copying or standard initialization with small perturbation) may be preferable.

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.

6 Implementation and Practical Considerations

6.1 Recommended Hyperparameters

Remark 18 (Hyperparameter Tuning)**.**

- *For small models (¡100M parameters): Use default values*
- *For large models (¿1B parameters): Reduce β_0, γ_0 by 30-50% for greater stability*
- *For tasks with high non-convexity (e.g., GANs): Increase $\tau_{crit}, \kappa_{crit}$ to avoid premature expansions*
- *For tasks with smooth landscapes (e.g., regression): Can increase β_0, γ_0 for faster convergence*

6.2 Computational Cost

Proposition 2 (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).

Remark 19 (Practical Optimizations). • ***Randomized SVD****: For very large P, use randomized algorithms that compute approximations of k singular vectors in $O(SP \cdot k \log k)$*

- ***Incremental computation****: Update SVD incrementally at each new checkpoint instead of recomputing from scratch*
- ***GPU****: All operations (norms, inner products, SVD) are parallelizable and efficient on GPU*

6.3 Memory Management

Remark 20 (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:

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

This is manageable with disk streaming or additional compression.

7 Extensions and Future Work

7.1 Potential Extensions

1. **Simultaneous multi-layer extrapolation**:
   - Analyze cross-covariance between trajectories of different layers
   - Grow multiple layers in parallel while maintaining global coherence
   - Use higher-order tensor analysis to capture interactions
2. **Online hyperparameter adaptation**:
   - Use meta-learning to adjust $\beta_0, \gamma_0, \eta_0$ during training

- Bayesian optimization of thresholds $\tau_{\text{crit}}, \kappa_{\text{crit}}$
- Policy learning for expansion decisions via reinforcement learning

3. **Integration with Neural Architecture Search**:
   - Combine AIether with NAS to decide not only when but where to add layers
   - Search for layer-specific extrapolation hyperparameters
   - Heterogeneous growth (different layer types at different positions)
4. **Analysis on non-Euclidean manifolds**:
   - Extend to parameters with special geometric structure (orthogonal matrices, simplexes)
   - Use Riemannian geometry to define intrinsic manifold metrics
   - Extrapolation on Stiefel/Grassmann manifolds for attention layers
5. **Combined pruning and growth**:
   - Alternate between growth (via AIether) and pruning (via magnitude pruning)
   - Maintain controlled complexity while exploring different architectures
   - Joint optimization of depth and width

7.2 Open Questions

1. **Rigorous theoretical analysis**: Develop provable convergence bounds under verifiable conditions (e.g., PL-condition, restricted strong convexity)
2. **Automatic choice of k**: Principles for selecting dynamic subspace dimension based on spectral properties (e.g., spectral gap, participation ratio)
3. **Generalization beyond Transformers**: Systematic validation on other architectures (RNNs, Graph NNs, Mixture of Experts)
4. **Cross-domain transfer**: Can trajectory from one domain inform initialization in a related domain?
5. **Extreme scale**: Behavior in models ¿100B parameters with limited computational resources

8 Conclusion

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

- **Geometric analysis**: Quantitative characterization of optimization trajectory via metrics τ, κ, ρ
- **Spectral decomposition**: Identification of dynamic subspaces via SVD to capture learning patterns
- **Adaptive extrapolation**: Informed initialization combining trend, curvature correction, and exploration
- **Stagnation detection**: Quantitative and persistent criteria to trigger architectural intervention

AIether targets three main objectives:

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

The approach is:

- **Tensor-wise**: Operates individually on each tensor, respecting specific geometric structures
- **Efficient**: Complexity $O(SPk)$ per layer, negligible vs. training cost
- **Scalable**: Applicable to large-scale models with appropriate compression strategies
- **Modular**: Can be integrated into existing training pipelines without architectural modifications

While extensive experimental validation is necessary, the rigorous mathematical foundation and detailed practical considerations establish a solid basis for future investigation. Implementation code and experiments will be made available in a public repository for reproducibility and 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] Chen, T., Goodfellow, I., & Shlens, J. (2015). Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

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

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

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

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

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

[8] 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_{final}

$M \leftarrow M_0$
$\mathcal{H}_{\text{layer}} \leftarrow \{\}$ ▷ Histories per layer
$c_{\text{stag}} \leftarrow 0$
while not converged **do**
    for $t = 1$ to $N_{\text{steps\_per\_eval}}$ **do**
        Train M for 1 step on $\mathcal{D}$
        if $t \bmod \Delta_{\text{collect}} = 0$ **then**
            Collect snapshot of frontier layer ℓ_{front}
            $\mathcal{H}_{\text{layer}}[\ell_{\text{front}}]$.append($W_{\ell_{\text{front}}}$)
        end if
    end for
    Evaluation and Stagnation Check:
    $\tau \leftarrow$ compute_TSF($\mathcal{H}_{\text{layer}}[\ell_{\text{front}}]$)
    $\kappa \leftarrow$ compute_curvature($\mathcal{H}_{\text{layer}}[\ell_{\text{front}}]$)
    if $\tau \geq \tau_{\text{crit}}$ **and** $\kappa \geq \kappa_{\text{crit}}$ **then**
        $c_{\text{stag}} \leftarrow c_{\text{stag}} + 1$
    else
        $c_{\text{stag}} \leftarrow 0$
    end if
    if $c_{\text{stag}} \geq p_{\text{patience}}$ **then**
        Trigger Growth:
        $W_{\text{new}} \leftarrow$ GeometricExtrapolator($\mathcal{H}_{\text{layer}}[\ell_{\text{front}}], \mathcal{H}$)
        $M \leftarrow$ GrowModel(M, W_{new})
        $c_{\text{stag}} \leftarrow 0$
        $\mathcal{H}_{\text{layer}}[\ell_{\text{front}} - 1] \leftarrow [\,]$ ▷ Initialize new history
    end if
end while
return M

B Implementation Details

B.1 TSF Computation Function

Algorithm 4 Compute TSF (Temporal Stretch Factor)

Require: History $\{W_0, \ldots, W_{S-1}\}$, $\varepsilon = 10^{-8}$
Ensure: $\tau \in [1, \infty)$

$\Delta \leftarrow W_{S-1} - W_0$
$\mathcal{L}_{\text{direct}} \leftarrow \|\Delta\|_F$
$\mathcal{L}_{\text{path}} \leftarrow 0$
for $t = 1$ to $S - 1$ **do**
    $V_t \leftarrow W_t - W_{t-1}$
    $\mathcal{L}_{\text{path}} \leftarrow \mathcal{L}_{\text{path}} + \|V_t\|_F$
end for
$\tau \leftarrow \mathcal{L}_{\text{path}}/(\mathcal{L}_{\text{direct}} + \varepsilon)$
return τ

B.2 Curvature Computation Function

Algorithm 5 Compute Curvature (Effective Curvature)

Require: History $\{W_0, \ldots, W_{S-1}\}$, $\varepsilon = 10^{-8}$
Ensure: $\kappa \in [0, \infty)$

$\Delta \leftarrow W_{S-1} - W_0$
$\hat{\Delta} \leftarrow \Delta/(\|\Delta\|_F + \varepsilon)$
sum_parallel $\leftarrow 0$
sum_perp $\leftarrow 0$
for $t = 1$ to $S - 1$ **do**
    $V_t \leftarrow W_t - W_{t-1}$
    $V_t^{\parallel} \leftarrow \langle V_t, \hat{\Delta} \rangle_F \cdot \hat{\Delta}$
    $V_t^{\perp} \leftarrow V_t - V_t^{\parallel}$
    sum_parallel $\leftarrow$ sum_parallel $+ \|V_t^{\parallel}\|_F$
    sum_perp $\leftarrow$ sum_perp $+ \|V_t^{\perp}\|_F$
end for
$\kappa \leftarrow$ sum_perp/(sum_parallel $+ \varepsilon$)
return κ