# Deep Delta Learning

January 1, 2026

**Abstract**

The efficacy of deep residual networks is fundamentally predicated on the identity shortcut connection. While this mechanism effectively mitigates the vanishing gradient problem, it imposes a strictly additive inductive bias on feature transformations, thereby limiting the network's capacity to model complex state transitions. In this paper, we introduce **Deep Delta Learning (DDL)**, a novel architecture that generalizes the standard residual connection by modulating the identity shortcut with a learnable, data-dependent geometric transformation. This transformation, termed the **Delta Operator**, constitutes a rank-1 perturbation of the identity matrix, parameterized by a reflection direction vector $\mathbf{k}(\mathbf{X})$ and a gating scalar $\beta(\mathbf{X})$. We provide a spectral analysis of this operator, demonstrating that the gate $\beta(\mathbf{X})$ enables dynamic interpolation between identity mapping, orthogonal projection, and geometric reflection. Furthermore, we restructure the residual update as a synchronous rank-1 injection, where the gate acts as a dynamic step size governing both the erasure of old information and the writing of new features. This unification empowers the network to explicitly control the spectrum of its layer-wise transition operator, enabling the modeling of complex, non-monotonic dynamics while preserving the stable training characteristics of gated residual architectures.

Project Page: `https://github.com/yifanzhang-pro/deep-delta-learning`

## 1 Introduction

Deep residual networks (He et al., 2016) represent a paradigm shift in neural network design, enabling the stable training of models with unprecedented depth. Their core mechanism, the identity shortcut connection, reformulates layers to learn a residual function $\mathbf{F}(\mathbf{X})$ with respect to their input $\mathbf{X}$. In its canonical form, the residual update is an element-wise addition:

$$\mathbf{X}_{l+1} = \mathbf{X}_l + \mathbf{F}(\mathbf{X}_l) \tag{1.1}$$

This formulation approximates a single forward Euler step for the matrix ordinary differential equation (ODE) $\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X})$, establishing a profound connection between deep learning and dynamical systems (Chen et al., 2018). However, this strictly additive nature imposes a strong translational prior on the learned dynamics; the Jacobian of the shortcut connection is rigidly fixed as the identity operator. Consequently, the network lacks an intrinsic mechanism to rotate, scale, or reflect learned feature matrices in a data-dependent manner without incurring prohibitive parameter costs in the residual function $\mathbf{F}(\cdot)$.

This structural rigidity inherently limits the expressive power of the network regarding state transitions. Recent work has highlighted the necessity of flexible transitions, particularly those capable of representing transformations with negative eigenvalues, for modeling complex phenomena such as oscillatory or oppositional patterns (Grazzi et al., 2024).

To overcome this limitation, we propose a principled generalization of the residual connection rooted in geometric linear algebra. We introduce **Deep Delta Learning (DDL)**, featuring a novel residual block that applies a learnable, rank-1 transformation to the hidden state matrix $\mathbf{X} \in \mathbb{R}^{d \times d_v}$. This formulation aligns the network depth with memory-augmented architectures, effectively treating the hidden state as a dynamic value matrix. This block utilizes a single learned scalar gate $\beta(\mathbf{X})$ to smoothly interpolate between a standard residual connection, an orthogonal projection operator, and a full geometric reflection. Our contributions are:

1. We propose the **Delta Residual Block**, a multi-branch architecture that learns to apply a generalized Householder operator to the matrix-valued shortcut connection, parameterized by a learned direction $\mathbf{k}(\mathbf{X})$ and a learned gate $\beta(\mathbf{X})$.

2. We provide a spectral analysis of the **Delta Operator**, deriving its complete eigensystem and demonstrating how the gate $\beta(\mathbf{X})$ deterministically controls the geometric nature of the transformation by shaping its spectrum.

3. We show that our formulation unifies identity mapping, projection, and reflection into a single, continuously differentiable module. We further demonstrate that DDL recovers the Delta Rule update mechanism, interpreting the gate $\beta$ as a dynamic step size for depth-wise optimization.

## 2 The Delta Residual Block

We build our method upon the mathematical foundation of the Householder reflection, which we generalize into a learnable, state-dependent operator.

### 2.1 Preliminaries: The Householder Transformation

**Definition 2.1** (Householder Matrix)**.** For a non-zero vector $\mathbf{k} \in \mathbb{R}^d$, the Householder matrix $\mathbf{H_k}$ is defined as:

$$\mathbf{H_k} = \mathbf{I} - 2\frac{\mathbf{k}\mathbf{k}^\top}{\|\mathbf{k}\|_2^2} \tag{2.1}$$

Geometrically, $\mathbf{H_k}$ reflects any vector across the hyperplane with normal vector $\mathbf{k}$.

The Householder matrix is a cornerstone of numerical linear algebra and possesses several key properties: it is symmetric ($\mathbf{H_k} = \mathbf{H_k}^\top$), orthogonal ($\mathbf{H_k}^\top \mathbf{H_k} = \mathbf{I}$), and involutory ($\mathbf{H_k}^2 = \mathbf{I}$). Its spectrum consists of a single eigenvalue of $-1$ (eigenvector $\mathbf{k}$) and $d-1$ eigenvalues of 1 (the eigenspace $\mathbf{k}^\perp$).

### 2.2 Formulation of the Delta Operator

We generalize the Householder matrix by replacing the constant factor of 2 with a learnable, data-dependent scalar gate, $\beta(\mathbf{X})$. This leads to the **Delta Residual (Delta-Res)** block. Let the hidden state be a matrix $\mathbf{X} \in \mathbb{R}^{d \times d_v}$, where $d$ is the feature dimension and $d_v$ denotes the number of value channels. We modify the additive residual to be a rank-1 update aligned with the reflection vector $\mathbf{k}$. The block output is computed as:

$$\mathbf{X}_{\text{out}} = \mathbf{A}(\mathbf{X}_{\text{in}})\mathbf{X}_{\text{in}} + \beta(\mathbf{X}_{\text{in}})\mathbf{k}(\mathbf{X}_{\text{in}})\mathbf{v}(\mathbf{X}_{\text{in}})^{\top} \tag{2.2}$$

where $\mathbf{v} \in \mathbb{R}^{d_v}$ is the **residual value vector** generated by the branch $\mathbf{F} : \mathbb{R}^{d \times d_v} \to \mathbb{R}^{d_v}$. Here, the outer product $\mathbf{k}\mathbf{v}^{\top}$ constitutes the additive update. Crucially, we apply the gate $\beta(\mathbf{X})$ to this constructive term as well, linking the erasure and write operations. The term $\mathbf{A}(\mathbf{X})$ is the **Delta Operator** acting spatially on the feature dimension $d$:

$$\mathbf{A}(\mathbf{X}) = \mathbf{I} - \beta(\mathbf{X})\frac{\mathbf{k}(\mathbf{X})\mathbf{k}(\mathbf{X})^{\top}}{\mathbf{k}(\mathbf{X})^{\top}\mathbf{k}(\mathbf{X}) + \epsilon} \tag{2.3}$$

The architecture learns the reflection direction $\mathbf{k}(\mathbf{X}) \in \mathbb{R}^d$, the value vector $\mathbf{v}(\mathbf{X}) \in \mathbb{R}^{d_v}$, and the reflection intensity $\beta(\mathbf{X}) \in \mathbb{R}$ through separate, lightweight neural network branches. The constant $\epsilon > 0$ ensures numerical stability. For the theoretical analysis, we assume $\mathbf{k}$ is strictly normalized such that $\mathbf{k}^{\top}\mathbf{k} = 1$ (see Appendix A for implementation details). Under this condition ($\epsilon \to 0$), the operator simplifies to:

$$\mathbf{A}(\mathbf{X}) = \mathbf{I} - \beta(\mathbf{X})\mathbf{k}(\mathbf{X})\mathbf{k}(\mathbf{X})^{\top} \tag{2.4}$$

Since $\mathbf{X}$ is a matrix, the operator $\mathbf{A}(\mathbf{X})$ broadcasts across the value dimension $d_v$, applying the geometric transformation simultaneously to every column of the hidden state.

The gating function $\beta(\mathbf{x})$ is parameterized to lie in the range $[0, 2]$ via a projection of the state features followed by a sigmoid function:

$$\beta(\mathbf{X}) = 2 \cdot \sigma(\text{Linear}(\mathcal{G}(\mathbf{X}))) \tag{2.5}$$

where $\mathcal{G}(\cdot)$ is a pooling, convolution, or flattening operation. This specific range is chosen for its rich geometric interpretations, which we analyze next.

## 3 Theoretical Analysis

The expressive power of the Delta-Res block stems from the spectral properties of the operator $\mathbf{A}(\mathbf{X})$, which are controlled by the learned gate $\beta(\mathbf{X})$.

### 3.1 Spectral Decomposition of the Delta Operator

**Theorem 3.1** (Spectrum of the Delta Operator)**.** Let $\mathbf{A} = \mathbf{I} - \beta\mathbf{k}\mathbf{k}^{\top}$ where $\mathbf{k} \in \mathbb{R}^d$ is a unit vector ($\mathbf{k}^{\top}\mathbf{k} = 1$) and $\beta \in \mathbb{R}$ is a scalar. The spectrum of $\mathbf{A}$, denoted $\sigma(\mathbf{A})$, is given by:

$$\sigma(\mathbf{A}) = \{\underbrace{1, 1, \ldots, 1}_{d-1 \text{ times}}, 1 - \beta\} \tag{3.1}$$

The eigenvector corresponding to the eigenvalue $\lambda = 1 - \beta$ is $\mathbf{k}$. The eigenspace for the eigenvalue $\lambda = 1$ is the orthogonal complement of $\mathbf{k}$, denoted $\mathbf{k}^{\perp} = \{\mathbf{u} \in \mathbb{R}^d \mid \mathbf{k}^{\top}\mathbf{u} = 0\}$.

*Proof.* Let $\mathbf{u}$ be any vector in the hyperplane orthogonal to $\mathbf{k}$ (i.e., $\mathbf{u} \in \mathbf{k}^\perp$ such that $\mathbf{k}^\top \mathbf{u} = 0$). Applying $\mathbf{A}$ to $\mathbf{u}$ yields:

$$\mathbf{A}\mathbf{u} = (\mathbf{I} - \beta \mathbf{k}\mathbf{k}^\top)\mathbf{u} = \mathbf{I}\mathbf{u} - \beta \mathbf{k}(\mathbf{k}^\top \mathbf{u}) = \mathbf{u} - \beta \mathbf{k}(0) = \mathbf{u} = 1 \cdot \mathbf{u} \tag{3.2}$$

Thus, any vector in the $(d-1)$-dimensional subspace $\mathbf{k}^\perp$ is an eigenvector with eigenvalue $\lambda = 1$.

Now, consider applying $\mathbf{A}$ to the vector $\mathbf{k}$ itself:

$$\mathbf{A}\mathbf{k} = (\mathbf{I} - \beta \mathbf{k}\mathbf{k}^\top)\mathbf{k} = \mathbf{I}\mathbf{k} - \beta \mathbf{k}(\mathbf{k}^\top \mathbf{k}) = \mathbf{k} - \beta \mathbf{k}(1) = (1 - \beta)\mathbf{k} \tag{3.3}$$

Thus, $\mathbf{k}$ is an eigenvector with eigenvalue $\lambda = 1 - \beta$. Since we have found $d$ linearly independent eigenvectors spanning $\mathbb{R}^d$, we have characterized the full spectrum of $\mathbf{A}$. □

This theorem provides a clear and powerful interpretation of the gate $\beta(\mathbf{X})$. By learning a single scalar, the network can dynamically control the geometry of the residual transformation across all $d_v$ columns of the state matrix simultaneously.

**Corollary 3.2** (Spatial Determinant)**.** The determinant of the Delta Operator $\mathbf{A}(\mathbf{X})$, acting on the spatial features $\mathbb{R}^d$, is given by:

$$\det(\mathbf{A}(\mathbf{X})) = \prod_{i=1}^{d} \lambda_i = 1^{d-1} \cdot (1 - \beta(\mathbf{X})) = 1 - \beta(\mathbf{X}) \tag{3.4}$$

This result indicates that $\beta(\mathbf{X})$ directly controls the orientation and volume of the state space transformation. For $\beta(\mathbf{X}) > 1$, the operator locally inverts the state matrix along the direction $\mathbf{k}(\mathbf{X})$, introducing a negative eigenvalue into the transition map.

## 3.2 Unification of Geometric Operations

Theorem 3.1 reveals that the range $[0, 2]$ for $\beta(\mathbf{X})$ allows the operator to interpolate between three fundamental linear transformations.

- **Identity Mapping ($\beta(\mathbf{X}) \to 0$):** As $\beta \to 0$, the eigenvalue $1 - \beta \to 1$. All eigenvalues of $\mathbf{A}(\mathbf{X})$ become 1, so $\mathbf{A}(\mathbf{X}) \to \mathbf{I}$. Since $\beta$ also modulates the injection term $\beta \mathbf{k}\mathbf{v}^\top$, the entire update vanishes, meaning $\mathbf{X}_{l+1} \approx \mathbf{X}_l$. This identity behavior is crucial for preserving signal propagation in very deep networks.
- **Orthogonal Projection ($\beta(\mathbf{X}) \to 1$):** As $\beta \to 1$, the eigenvalue $1 - \beta \to 0$. The operator $\mathbf{A}(\mathbf{X})$ becomes $\mathbf{I} - \mathbf{k}\mathbf{k}^\top$, which is a projection matrix that maps vectors onto the hyperplane $\mathbf{k}^\perp$. The component of the input state $\mathbf{X}$ parallel to $\mathbf{k}$ is explicitly removed ("forgotten") before the residual is added. The operator becomes singular, and $\det(\mathbf{A}) \to 0$.
- **Full Reflection ($\beta(\mathbf{X}) \to 2$):** As $\beta \to 2$, the eigenvalue $1 - \beta \to -1$. The operator $\mathbf{A}(\mathbf{X})$ becomes $\mathbf{I} - 2\mathbf{k}\mathbf{k}^\top$, a standard Householder matrix. This performs a perfect reflection of each column of $\mathbf{X}$ across $\mathbf{k}^\perp$. This is the only case in this range where the transformation is guaranteed to be orthogonal and spatially volume-preserving, with $\det(\mathbf{A}) \to -1$. The negative spatial determinant signifies a change in orientation (a reflection) of the basis.

## 3.3 Special Case: Gated Residual Learning

A critical property of Deep Delta Learning is its behavior in the limit of the gating scalar. When the gate vanishes ($\beta(\mathbf{X}) \to 0$), the Delta Operator converges to the identity matrix ($\mathbf{A}(\mathbf{X}) \to \mathbf{I}$), and the constructive term vanishes. Consequently, the update rule in Equation (2.2) simplifies to:

$$\mathbf{X}_{l+1} = \mathbf{X}_l \tag{3.5}$$

This recovers the identity mapping, effectively allowing the layer to be skipped entirely. This behavior is consistent with the zero-initialization strategy often required for training very deep networks. Conversely, when $\beta \approx 1$, the layer functions as a Gated Rank-1 Matrix ResNet, where $\beta$ acts as a learned step size governing the magnitude of the update. This demonstrates that DDL generalizes residual learning by introducing a multiplicative, geometric modulation that is coupled synchronously with the value injection.

## 3.4 Case Study: Interaction with Diagonal Feature Matrices

To better understand the mixing properties of the Delta Operator, consider the special case where the input state $\mathbf{X} \in \mathbb{R}^{d \times d}$ is a square diagonal matrix, $\mathbf{X} = \operatorname{diag}(\lambda_1, \ldots, \lambda_d)$. This represents a state where features are perfectly decoupled across the value dimensions. The application of $\mathbf{A}$ yields:

$$(\mathbf{A}\mathbf{X})_{ij} = (\mathbf{X} - \beta \mathbf{k}\mathbf{k}^\top \mathbf{X})_{ij} = \lambda_i \delta_{ij} - \beta \lambda_j k_i k_j \tag{3.6}$$

Specifically, the off-diagonal element ($i \neq j$) becomes $-\beta \lambda_j k_i k_j$, while the diagonal element ($i = j$) is scaled to $\lambda_i(1 - \beta k_i^2)$. This implies that the output feature $i$ is now dependent on the magnitude of the input feature $j$, scaled by the geometric coherence $k_i k_j$. This result elucidates a critical function of the Delta block: it induces controlled feature coupling. Even if the incoming features are independent, a non-zero $\beta$ forces an interaction between the $i$-th and $j$-th modes proportional to the projection of the reflection vector $\mathbf{k}$.

If $\beta \to 1$ (projection), the operator subtracts the weighted correlations, effectively orthogonalizing the state with respect to the direction $\mathbf{k}$. If $\beta \to 0$, the diagonal structure is preserved.

## 3.5 Vector Hidden State Dynamics

While DDL operates on matrix-valued states $\mathbf{X} \in \mathbb{R}^{d \times d_v}$, it naturally encapsulates standard vector-based deep learning as a specific limit. We identify two distinct regimes:

**The Scalar Value Limit ($d_v = 1$).** When the value dimension is reduced to unity, the hidden state degenerates to a standard feature vector $\mathbf{x} \in \mathbb{R}^d$. In this limit, the value update $\mathbf{v}$ becomes a scalar $v \in \mathbb{R}$. The Delta update rule (2.2) simplifies to:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \beta_l \underbrace{(v_l - \mathbf{k}_l^\top \mathbf{x}_l)}_{\gamma_l} \mathbf{k}_l \tag{3.7}$$

Here, the geometric reflection collapses into a dynamic scalar gating mechanism. The term $\gamma_l$ acts as a data-dependent coefficient that couples the residual magnitude to the error between the target value $v_l$ and the projection of the current state $\mathbf{x}_l$ onto $\mathbf{k}_l$.

**The Independent Feature Limit.** Alternatively, one may view the diagonal case in Section 3.4 as a representation of a vector state embedded in a matrix diagonal. As shown in the diagonal analysis, the Delta Operator introduces feature coupling via the term $\beta k_i k_j$. To recover the behavior of standard element-wise vector updates (where features do not mix spatially), the reflection vector $\mathbf{k}$ must be aligned with the canonical basis (i.e., one-hot). In this regime, the Delta Operator acts as an element-wise gating function, strictly preserving the independence of the feature dimensions.

## 4 Connections to Optimization and Delta Architectures

The terminology Deep Delta Learning reflects a structural homology with the Delta Rule, a fundamental update mechanism recently popularized in efficient sequence modeling, e.g., DeltaNet (Schlag et al., 2021; Yang et al., 2024).

### 4.1 The Delta Rule for Residual Learning

The standard residual connection, $\mathbf{X}_{l+1} = \mathbf{X}_l + \mathbf{F}(\mathbf{X}_l)$, imposes a strictly additive inductive bias. Information, once generated by $\mathbf{F}$, is simply accumulated. This can lead to "residual accumulation", where noisy or interfering features persist across layers because the network lacks an explicit mechanism to selectively filter the hidden state.

Deep Delta Learning addresses this by incorporating the Delta Rule structure into the depth dimension. Expanding the Delta Residual update in Equation (2.2) using the rank-1 residual definition:

$$\mathbf{X}_{l+1} = \mathbf{X}_l + \beta_l \mathbf{k}_l \left( \underbrace{\mathbf{v}_l^\top}_{\text{Write}} - \underbrace{\mathbf{k}_l^\top \mathbf{X}_l}_{\text{Erase}} \right) \tag{4.1}$$

This formulation exactly recovers the Delta Rule update utilized in fast associative memories and linear attention. The term $\mathbf{k}_l^\top \mathbf{X}_l$ represents the current projection of the state onto the reflection vector (the "error" or "old memory"). The term $(\mathbf{v}_l^\top - \mathbf{k}_l^\top \mathbf{X}_l)$ acts as the correction signal.

Since $\mathbf{X}_l \in \mathbb{R}^{d \times d_v}$ is a matrix, the term $\mathbf{k}_l^\top \mathbf{X}_l$ yields a row vector in $\mathbb{R}^{1 \times d_v}$, representing the projection of *every* value column onto $\mathbf{k}_l$. The update rigidly aligns both the erasure (destructive) and injection (constructive) operations along the geometric direction defined by the projector $\mathbf{k}_l$, modulated by the step size $\beta_l$.

When $\beta(\mathbf{X}_l) \approx 1$, this subtractive term acts as an orthogonal projection, effectively erasing the component of the incoming state $\mathbf{X}_l$ parallel to $\mathbf{k}(\mathbf{X}_l)$ (forgetting). When $\beta(\mathbf{X}_l) \approx 2$, the term subtracts twice the projection, resulting in a sign inversion (reflection). This provides the network with a flexible mechanism to selectively clean or reorient specific feature subspaces layer-by-layer, preventing the accumulation of interference.

### 4.2 Relation to DeltaNets and Householder Products

Our work shares a theoretical link with the **DeltaNet** architecture (Schlag et al., 2021), which replaces the additive accumulation of Linear Transformers with a "Delta Rule" for memory updates.

We demonstrate that Deep Delta Learning is the *depth-wise isomorphism* of the DeltaNet recurrence. In DeltaNet, the hidden state (memory) $\mathbf{S}_t$ evolves over time $t$. To unify notation with our depth-wise formulation, we present the DeltaNet update using left-multiplication semantics, where the memory state is $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$:

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top \tag{4.2}$$

Here, the operator acts on the key dimension $d_k$, which is analogous to the feature dimension $d$ in DDL. Comparing this to our Deep Delta Layer update Equation (2.2) acting over depth $l$:

$$\mathbf{X}_{l+1} = (\mathbf{I} - \beta_l \mathbf{k}_l \mathbf{k}_l^\top)\mathbf{X}_l + \beta_l \mathbf{k}_l \mathbf{v}_l^\top \tag{4.3}$$

where $\mathbf{v}_l$ is the vector output of the value branch.

This reveals a direct structural correspondence:

- **State Transition:** The memory state $\mathbf{S}_t$ (dimension $d_k \times d_v$) in DeltaNet corresponds to the feature activation $\mathbf{X}_l$ (dimension $d \times d_v$) in DDL.
- **Geometric Filtering:** Both architectures employ the rank-1 Householder operator to selectively reflect or erase subspace components. DeltaNet applies this over time steps $t$, whereas DDL applies it over network depth $l$.
- **Rank-1 Injection:** Our modified residual update $\beta_l \mathbf{k}_l \mathbf{v}_l^\top$ aligns perfectly with the DeltaNet write operation. By incorporating $\beta_l$ into the constructive term, we interpret $\beta_l$ as a layer-wise step size for the depth-wise ODE. This ensures that both the erasure (destructive) and injection (constructive) components are modulated synchronously, ensuring the update represents a coherent geometric transformation of the state $\mathbf{X}$.

Thus, DDL can be interpreted as applying the "Delta Rule" to layer-wise feature evolution, enabling the network to "forget" or "rewrite" features from shallow layers as they propagate deeper.

## 5 Related Work

Our work builds upon several key research themes in deep learning.

**Gated and Invertible Architectures.** Highway Networks (Srivastava et al., 2015) introduced data-dependent gating to residual networks, but their gates interpolate between the identity path and the function path, rather than modifying the transformation itself. Invertible Residual Networks (i-ResNets) (Behrmann et al., 2019) constrain the Lipschitz constant of $\mathbf{F}$ to ensure invertibility, which is useful for applications like normalizing flows. Our Delta Operator is involutory (and thus invertible) when $\beta = 2$, but it does not enforce this property, instead allowing the network to learn when and where invertibility is useful.

**Orthogonal and Unitary Networks.** A significant body of work has focused on constraining network weights to be orthogonal or unitary to improve gradient stability and preserve geometric structure (Arjovsky et al., 2016; Jing et al., 2017). Householder reflections are a classic method for parameterizing orthogonal matrices. These methods enforce orthogonality as a strict constraint. In contrast, our Delta Residual Network learns to deviate from identity and orthogonality via the gate $\beta(\mathbf{x})$, providing a soft, adaptive constraint that can be relaxed to pure projection or even reflection.

**Neural Ordinary Differential Equations.** Neural ODEs (Chen et al., 2018) model the continuous evolution of features. The standard ResNet Eq. (1.1) is a discretization of the simple ODE $\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X})$. Our proposed architecture alters the underlying dynamics to $\dot{\mathbf{X}} = \beta(\mathbf{X})\mathbf{k}(\mathbf{X})(\mathbf{v}(\mathbf{X})^\top - \mathbf{k}(\mathbf{X})^\top\mathbf{X})$, introducing a state-dependent projection term applied to the matrix state. This allows for a much richer family of learnable dynamical systems that can exhibit contractive or oscillatory behavior across multiple value dimensions.

## 6 Conclusion

We have introduced Deep Delta Learning, a novel architecture built upon an adaptive, geometric residual connection. Through analysis, we have demonstrated that its core component, the Delta Operator, unifies identity mapping, projection, and reflection into a single, continuously differentiable module. This unification is controlled by a simple learned scalar gate, which dynamically shapes the spectrum of the layer-to-layer transition operator. By empowering the network to learn transformations with negative eigenvalues in a data-dependent fashion, DDL offers a significant and principled increase in expressive power while retaining the foundational benefits of the residual learning paradigm.

## References

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International conference on machine learning*, pages 573–582. PMLR, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. *arXiv preprint arXiv:2411.12537*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *International Conference on Machine Learning*, pages 1733–1741. PMLR, 2017.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.

# Appendix

## A Implementation and Parameterization Details

The Deep Delta Learning (DDL) framework relies on the efficient estimation of the reflection direction $\mathbf{k}(\mathbf{X})$, the scalar gate $\beta(\mathbf{X})$, and the residual value $\mathbf{v}(\mathbf{X})$. While the theoretical results hold regardless of the specific topology used to approximate these functions, we outline two primary architectural instantiations for the generator functions: MLP-based and Attention-based parameterizations.

Let the hidden state be $\mathbf{X} \in \mathbb{R}^{d \times d_v}$. We denote the generator branch for the reflection vector as a function $\phi_k : \mathbb{R}^{d \times d_v} \to \mathbb{R}^d$.

### A.1 Parameterization of the Reflection Direction $\mathbf{k}(\mathbf{X})$

The geometric orientation of the Delta Operator is determined by $\mathbf{k}$. We propose two distinct mechanisms for $\phi_k$, allowing for different inductive biases regarding feature interaction.

**Option 1: MLP Parameterization.** For architectures prioritizing global feature mixing with low computational overhead, we parameterize $\mathbf{k}$ using a Multi-Layer Perceptron (MLP) acting on aggregated statistics of the state matrix.

$$\widetilde{\mathbf{k}}_{\text{MLP}} = \text{MLP}\left(\text{Pool}(\mathbf{X})\right), \quad \mathbf{k}_{\text{MLP}} = \frac{\widetilde{\mathbf{k}}_{\text{MLP}}}{\|\widetilde{\mathbf{k}}_{\text{MLP}}\|_2} \tag{A.1}$$

Here, $\text{Pool}(\cdot)$ represents a reduction operation (e.g., Global Average Pooling or Flattening) that maps $\mathbb{R}^{d \times d_v} \to \mathbb{R}^d$. We enforce $L_2$ normalization to satisfy the spectral assumptions in Theorem 3.1.

**Option 2: Attention-based Parameterization.** To capture more granular dependencies within the value dimension, we can employ attention mechanism.

### A.2 Parameterization of the Gate $\beta(\mathbf{X})$ and Value $\mathbf{v}(\mathbf{X})$

**The Gating Branch.** The scalar gate $\beta$ requires a bounded output in $[0, 2]$. We maintain a lightweight design for this estimator:

$$\beta(\mathbf{X}) = 2 \cdot \sigma\left(\mathbf{w}_\beta^\top \tanh(\mathbf{W}_{\text{in}} \text{Pool}(\mathbf{X}))\right) \tag{A.2}$$

where $\sigma$ is the sigmoid function, ensuring smooth interpolation between identity, projection, and reflection.

**The Value Branch.** The residual value vector $\mathbf{v} \in \mathbb{R}^{d_v}$ represents the content update. This branch, $\mathbf{F} : \mathbb{R}^{d \times d_v} \to \mathbb{R}^{d_v}$, allows for flexible design choices. In our experiments, we utilize the same architecture chosen for the main backbone (e.g., if DDL is applied in a Transformer, $\mathbf{F}$ mirrors the Feed-Forward Network or Multi-Head Attention block structure) to ensure capacity alignment.