Deep Delta Learning

Yifan Zhang[1] **Yifeng Liu**[2] **Mengdi Wang**[1] **Quanquan Gu**[2]
[1]Princeton University [2]University of California, Los Angeles
yifzhang@princeton.edu

January 1, 2026*

Abstract

The effectiveness of deep residual networks hinges on the identity shortcut connection. While this mechanism alleviates the vanishing-gradient problem, it also has a strictly additive inductive bias on feature transformations, limiting the network's ability to model complex hidden state transitions. In this paper, we introduce **Deep Delta Learning (DDL)**, which generalizes the shortcut from a fixed identity map to a learnable, state-dependent linear operator. The resulting Delta Operator is a rank-1 perturbation of the identity, $\mathbf{A}(\mathbf{X}) = \mathbf{I} - \beta(\mathbf{X})\mathbf{k}(\mathbf{X})\mathbf{k}(\mathbf{X})^\top$, parameterized by a unit direction $\mathbf{k}(\mathbf{X})$ and a scalar gate $\beta(\mathbf{X})$. We provide a spectral analysis showing that $\beta(\mathbf{X})$ continuously interpolates the shortcut between identity ($\beta = 0$), orthogonal projection ($\beta = 1$), and Householder reflection ($\beta = 2$). Furthermore, we rewrite the residual update as a synchronized rank-1 delta write: β scales both the removal of the current $\mathbf{k}$-component and the injection of the new $\mathbf{k}$-component. This unification enables explicit control of the shortcut spectrum along a data-dependent direction while retaining stable training behavior. Empirically, replacing Transformer residual additions with DDL improves validation loss and perplexity, as well as downstream evaluation accuracy on language modeling tasks, with larger gains in the expanded-state setting.

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 by enabling the stable training of models with unprecedented depth. Their core mechanism, the identity shortcut connection, reformulates each layer to learn a residual function $\mathbf{F}(\mathbf{X})$ with respect to its input $\mathbf{X}$. In its canonical form, the residual update is an element-wise addition:

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

We can view this as a forward Euler step for the ODE $\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X})$. This perspective connects deep networks to dynamical systems (Chen et al., 2018; Tong et al., 2025; Li et al., 2022; Kan et al., 2025; Jemley, 2026). The strictly additive update also imposes a strong *translational* bias on the learned dynamics. The shortcut path keeps a fixed Jacobian equal to the identity map.

*Revised: January 29, 2026

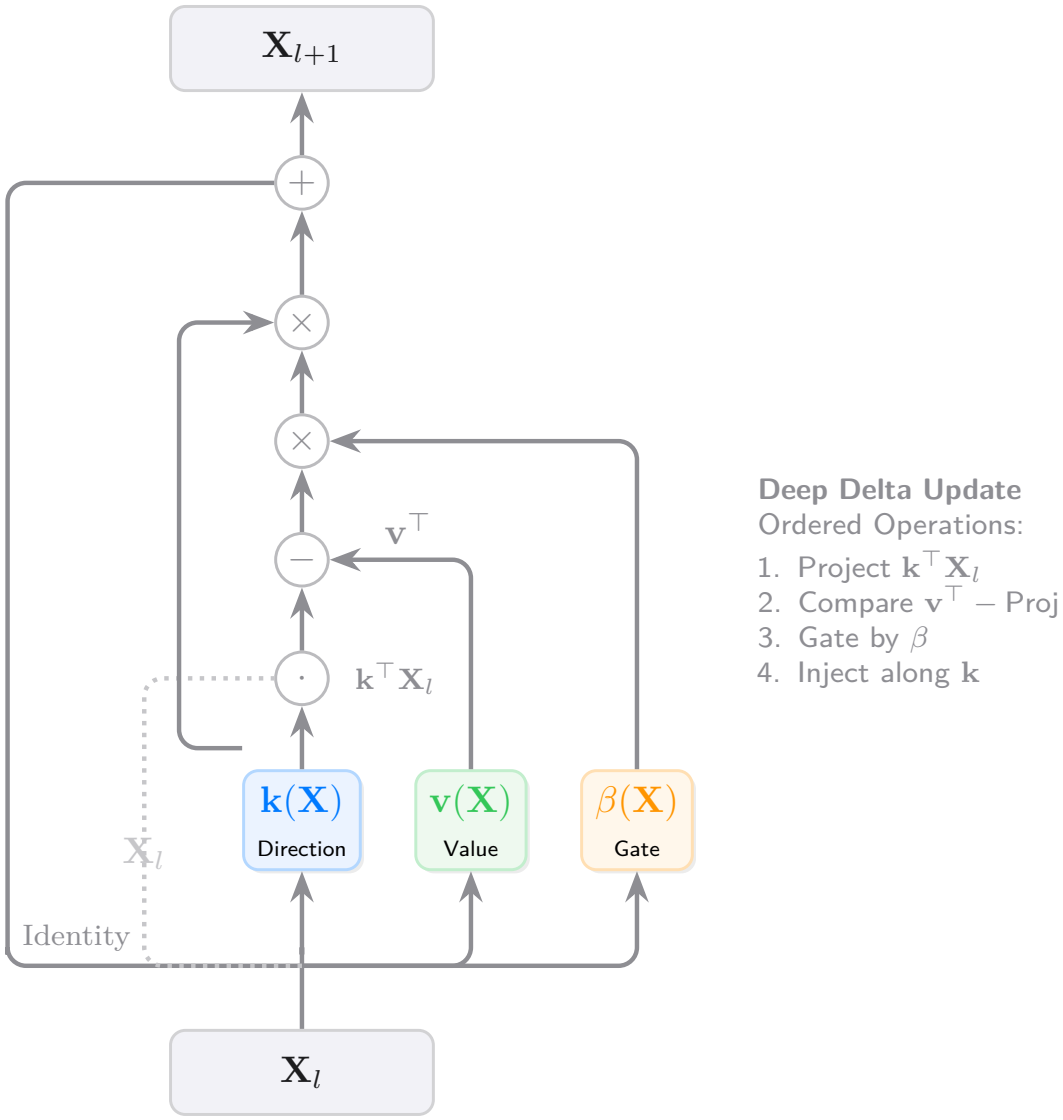


Figure 1 The Deep Delta Residual Block. The architecture generalizes the standard residual connection. A learnable scalar gate β controls a rank-1 geometric transformation.

This perspective also highlights a structural bias: the skip connection contributes an identity term to each layer's Jacobian, biasing the layer-wise linearization toward eigenvalues near +1. This can make certain transitions, notably those requiring negative eigenvalues along some directions, harder to realize without large residual corrections (Grazzi et al., 2025). Related lines of work have explored alternatives to fixed additive skip connections and to expanding the size of residual streams, including Alternating Updates (AltUp) (Baykal et al., 2023) and Hyper-Connections (Zhu et al., 2025). In contrast, DDL retains a single-stream residual formulation while endowing the shortcut with an explicitly analyzable, data-dependent spectrum via a low-rank operator.

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 residual block that applies a state-dependent rank-1 perturbation of the identity to the shortcut and a

synchronized rank-1 write into the hidden-state matrix $\mathbf{X} \in \mathbb{R}^{d \times d_v}$. This formulation aligns depth with memory-augmented architectures by treating the hidden state as a dynamic value matrix. Crucially, the learned scalar gate $\beta(\mathbf{X})$ interpolates the shortcut operator between the identity map, orthogonal projection, and (at $\beta = 2$) geometric reflection, while synchronously scaling the write term. Our contributions are:

1. We propose the **Delta Residual Block**, a multi-branch architecture that learns to apply a gated (generalized) Householder-style operator to the matrix-valued shortcut connection, parameterized by a learned direction $\mathbf{k}(\mathbf{X})$, a residual value vector $\mathbf{v}(\mathbf{X})$, and a learned gate $\beta(\mathbf{X})$, illustrated in Figure 1.
2. We give a spectral analysis of the Delta Operator. We derive its complete eigensystem and show how $\beta(\mathbf{X})$ controls the transformation by shaping its spectrum.
3. We unify identity mapping, projection, and reflection in one continuously differentiable module. We also show DDL recovers the Delta Rule update, with the gate β acting like a depth-wise step size.
4. We evaluate DDL as a drop-in replacement for Transformer residual additions on language modeling tasks at 124M and 353M scales, showing consistent improvements in loss, perplexity, and downstream benchmark evaluations, expanding the residual state to $d_v = 4$ yields the strongest gains.

2 Deep Delta Learning

We build our method on 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 (Householder, 1958))**.** For a nonzero vector $\mathbf{k} \in \mathbb{R}^d$, the Householder matrix $\mathbf{H}_\mathbf{k}$ is defined as:

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

Geometrically, $\mathbf{H}_\mathbf{k}$ reflects any vector across the hyperplane whose normal vector is $\mathbf{k}$.

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

2.2 Formulation of the Delta Operator

A standard Householder reflection can be written in normalized form as $\mathbf{H}_\mathbf{u} = \mathbf{I} - 2\mathbf{u}\mathbf{u}^\top$ for any unit vector $\mathbf{u}$. We generalize this construction in two ways: (i) we replace the constant factor 2 with a learnable, data-dependent scalar gate $\beta(\mathbf{X})$, and (ii) we let the normal direction depend on the current state via a learned map $\mathbf{k}(\mathbf{X})$. 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 into a rank-1 update aligned with the direction $\mathbf{k}$.

Normalized direction. Let $\widetilde{\mathbf{k}}(\mathbf{X}) \in \mathbb{R}^d$ be an unnormalized direction produced by a lightweight branch. For the analysis, we work with the unit direction

$$\mathbf{k}(\mathbf{X}) = \frac{\widetilde{\mathbf{k}}(\mathbf{X})}{\|\widetilde{\mathbf{k}}(\mathbf{X})\|_2}, \qquad \widetilde{\mathbf{k}}(\mathbf{X}) \neq \mathbf{0}, \tag{2.2}$$

so that $\mathbf{k}(\mathbf{X})^\top \mathbf{k}(\mathbf{X}) = 1$.

Delta Operator. The term $\mathbf{A}(\mathbf{X})$ is the Delta Operator acting spatially on the feature dimension d:

$$\mathbf{A}(\mathbf{X}) = \mathbf{I} - \beta(\mathbf{X})\, \mathbf{k}(\mathbf{X})\mathbf{k}(\mathbf{X})^\top. \tag{2.3}$$

Conditioned on the current state (and therefore on $\beta(\mathbf{X})$ and $\mathbf{k}(\mathbf{X})$), $\mathbf{A}(\mathbf{X})$ is a linear map on the spatial feature space $\mathbb{R}^d$. The overall layer mapping $\mathbf{X} \mapsto \mathbf{A}(\mathbf{X})\mathbf{X}$ is nonlinear because β and $\mathbf{k}$ (and $\mathbf{v}$ below) are functions of $\mathbf{X}$.

Delta-Res block output. The block output is computed as

$$\mathbf{X}_{l+1} = \mathbf{A}(\mathbf{X}_l)\mathbf{X}_l + \beta(\mathbf{X}_l)\mathbf{k}(\mathbf{X}_l)\mathbf{v}(\mathbf{X}_l)^\top, \tag{2.4}$$

where $\mathbf{v}(\mathbf{X}_l) \in \mathbb{R}^{d_v}$ is the residual value vector produced by a value branch $\phi_v : \mathbb{R}^{d \times d_v} \to \mathbb{R}^{d_v}$ (i.e., $\mathbf{v}(\mathbf{X}) := \phi_v(\mathbf{X})$). Here, the outer product $\mathbf{k}\mathbf{v}^\top$ constitutes the rank-1 write. Crucially, we apply the same gate $\beta(\mathbf{X})$ to this constructive term as well, synchronizing the erasure and the write.

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.

Substituting Eq. (2.3) into Eq. (2.4) yields an exact additive, rank-1 Delta form:

$$\mathbf{X}_{l+1} = \mathbf{X}_l + \beta(\mathbf{X}_l)\, \mathbf{k}(\mathbf{X}_l)\Big(\mathbf{v}(\mathbf{X}_l)^\top - \mathbf{k}(\mathbf{X}_l)^\top \mathbf{X}_l\Big), \tag{2.5}$$

which makes explicit that the same scalar β synchronizes (i) removal of the current $\mathbf{k}$-component, via the rank-1 term $\mathbf{k}(\mathbf{k}^\top \mathbf{X}_l) = (\mathbf{k}\mathbf{k}^\top)\mathbf{X}_l$, and (ii) injection of a new $\mathbf{k}$-component, via $\mathbf{k}\mathbf{v}^\top$. Under the unit-norm convention $\mathbf{k}(\mathbf{X})^\top \mathbf{k}(\mathbf{X}) = 1$ (Eq. (2.2)), $\mathbf{k}(\mathbf{X}_l)^\top \mathbf{X}_l \in \mathbb{R}^{1 \times d_v}$ is a row vector whose j-th entry equals the orthogonal-projection coefficient of the j-th value column of $\mathbf{X}_l$ onto $\mathbf{k}(\mathbf{X}_l)$.

Projected dynamics along the chosen direction. Projecting Eq. (2.5) onto the direction used at layer l yields a closed form:

$$\mathbf{k}_l^\top \mathbf{X}_{l+1} = (1 - \beta_l)\, \mathbf{k}_l^\top \mathbf{X}_l + \beta_l\, \mathbf{v}_l^\top, \tag{2.6}$$

where $\beta_l = \beta(\mathbf{X}_l)$, $\mathbf{k}_l = \mathbf{k}(\mathbf{X}_l)$, and $\mathbf{v}_l = \mathbf{v}(\mathbf{X}_l)$. In particular, when $\beta_l = 1$ the block overwrites the $\mathbf{k}_l$-component of each value column: $\mathbf{k}_l^\top \mathbf{X}_{l+1} = \mathbf{v}_l^\top$.

The gating function $\beta(\mathbf{X})$ is parameterized to lie in $(0, 2)$ by applying a sigmoid to a scalar logit:

$$\beta(\mathbf{X}) = 2 \cdot \sigma(\mathrm{Linear}(\mathcal{G}(\mathbf{X}))), \tag{2.7}$$

where $\mathcal{G}(\cdot)$ is a pooling, convolution, or flattening operation. The endpoint behaviors $\beta \to 0$ and $\beta \to 2$ are approached as the logit saturates; throughout, we interpret the cases $\beta = 0$ and $\beta = 2$ as these limits. This specific range is chosen for its rich geometric interpretations, which we analyze next.

3 Analysis of DDL

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 $\beta(\mathbf{X})$. For notational convenience, we fix a layer l and write $\beta_l := \beta(\mathbf{X}_l)$, $\mathbf{k}_l := \mathbf{k}(\mathbf{X}_l)$, and $\mathbf{v}_l := \mathbf{v}(\mathbf{X}_l)$; when analyzing the spectrum we suppress the layer index and treat β and $\mathbf{k}$ as fixed.

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 eigenvalues of $\mathbf{A}$ are 1 with multiplicity $d-1$ and $1-\beta$ with multiplicity 1. An eigenvector corresponding to the eigenvalue $1-\beta$ is $\mathbf{k}$. The eigenspace for the eigenvalue 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 $\mathbf{u}$ gives:

$$\begin{aligned}\mathbf{A}\mathbf{u} &= (\mathbf{I} - \beta\mathbf{k}\mathbf{k}^\top)\mathbf{u} = \mathbf{I}\mathbf{u} - \beta\mathbf{k}(\mathbf{k}^\top\mathbf{u}) \\ &= \mathbf{u} - \beta\mathbf{k}(0) = \mathbf{u} = 1 \cdot \mathbf{u}.\end{aligned}$$

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:

$$\begin{aligned}\mathbf{A}\mathbf{k} &= (\mathbf{I} - \beta\mathbf{k}\mathbf{k}^\top)\mathbf{k} = \mathbf{I}\mathbf{k} - \beta\mathbf{k}(\mathbf{k}^\top\mathbf{k}) \\ &= \mathbf{k} - \beta\mathbf{k}(1) = (1-\beta)\mathbf{k}.\end{aligned}$$

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 $\mathbf{A}$. □

Equivalently, for any $\mathbf{u} \in \mathbb{R}^d$ decompose $\mathbf{u} = \mathbf{u}_\perp + (\mathbf{k}^\top\mathbf{u})\mathbf{k}$ with $\mathbf{u}_\perp \in \mathbf{k}^\perp$. Then

$$\mathbf{A}\mathbf{u} = \mathbf{u}_\perp + (1-\beta)\,(\mathbf{k}^\top\mathbf{u})\mathbf{k},$$

making explicit that $\mathbf{A}$ leaves $\mathbf{k}^\perp$ unchanged and scales the $\mathbf{k}$-component by $(1-\beta)$.

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.

Lifting to matrix-valued states. The spectral statements above are spatial: they describe the linear map $\mathbf{u} \mapsto \mathbf{A}\mathbf{u}$ on $\mathbb{R}^d$. Since our hidden state is a matrix $\mathbf{X} \in \mathbb{R}^{d \times d_v}$ and the shortcut acts by left-multiplication, each of the d_v columns is transformed independently by the same $\mathbf{A}$. Equivalently, under vectorization, the induced linear operator is $\mathbf{I}_{d_v} \otimes \mathbf{A}$. Thus, the spectrum of the lifted map consists of the eigenvalues of $\mathbf{A}$ repeated d_v times, and its determinant equals $\det(\mathbf{A})^{d_v}$.

Because $\mathbf{A}$ is symmetric, its singular values are the absolute values of its eigenvalues, i.e., 1 (with multiplicity $d-1$) and $|1-\beta|$. In particular, under the unit-norm assumption, $\mathbf{A}$ is orthogonal if and only if $|1-\beta| = 1$, equivalently $\beta \in \{0, 2\}$. With the sigmoid parameterization in Eq. (2.7), these endpoint cases are approached in the saturated-logit limit. For $\beta \in (0, 2)$, $\mathbf{A}$ contracts the $\mathbf{k}$-component in magnitude (and flips its sign when $\beta > 1$) while leaving $\mathbf{k}^\perp$ unchanged.

Corollary 3.2 (Spatial Determinant)**.** Assume the unit-norm condition $\mathbf{k}(\mathbf{X})^\top \mathbf{k}(\mathbf{X}) = 1$ (Eq. (2.2)). The determinant of the Delta Operator $\mathbf{A}(\mathbf{X})$, acting on the spatial features $\mathbb{R}^d$, is:

$$\det(\mathbf{A}(\mathbf{X})) = 1 - \beta(\mathbf{X}).$$

Since the shortcut broadcasts across the d_v value columns, the induced determinant on the full matrix state space $\mathbb{R}^{d \times d_v}$ is $\det(\mathbf{A}(\mathbf{X}))^{d_v} = (1 - \beta(\mathbf{X}))^{d_v}$. Thus $\beta(\mathbf{X})$ controls the signed volume scaling along the distinguished spatial direction $\mathbf{k}(\mathbf{X})$, in particular, for $\beta(\mathbf{X}) \in (1, 2]$ the spatial eigenvalue $1 - \beta(\mathbf{X})$ is negative, causing a sign reversal along $\mathbf{k}(\mathbf{X})$, with a full Householder reflection occurring at $\beta(\mathbf{X}) = 2$. When the determinant is nonzero, the global orientation of the lifted state space flips if and only if d_v is odd and $\beta(\mathbf{X}) > 1$.

3.2 Unification of Geometric Operations

Theorem 3.1 reveals that as $\beta(\mathbf{X})$ moves from 0 to 2, the shortcut operator interpolates 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 β 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 $\mathbf{I} - \mathbf{k}\mathbf{k}^\top$, an orthogonal projector (rank $d - 1$) onto the hyperplane $\mathbf{k}^\perp$. The component of each column of the input state $\mathbf{X}$ parallel to $\mathbf{k}$ is explicitly removed before the residual is added. The operator becomes singular, and $\det(\mathbf{A}) \to 0$. In terms of the full block, this regime can be interpreted as replace-along-$\mathbf{k}$: the shortcut removes the $\mathbf{k}$-component, and the rank-1 write injects a new component along $\mathbf{k}$ specified by $\mathbf{v}^\top$. Eq. (2.6) makes this explicit: at $\beta_l = 1$, $\mathbf{k}_l^\top \mathbf{X}_{l+1} = \mathbf{v}_l^\top$.
- **Full Reflection ($\beta(\mathbf{X}) \to 2$):** As $\beta \to 2$, the eigenvalue $1 - \beta \to -1$ and $\mathbf{A}(\mathbf{X}) \to \mathbf{I} - 2\mathbf{k}\mathbf{k}^\top$, the standard Householder reflector. It reflects each column of $\mathbf{X}$ across $\mathbf{k}^\perp$. Like the identity case ($\beta \to 0$), this is an orthogonal, volume-preserving shortcut (all singular values equal to 1); here $\det(\mathbf{A}) \to -1$, indicating orientation reversal. The full block then adds the synchronized rank-1 write term $\beta \mathbf{k}\mathbf{v}^\top$.

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 Eq. (2.4) simplifies to:

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

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 β 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. More generally, Eq. (2.5) is a rank-1 delta update with step size β; the regime $\beta \approx 1$ corresponds to an overwrite-along-$\mathbf{k}$ behavior in the projected coordinates (Eq. (2.6)).

3.4 Diagonal Feature Matrices Case

To isolate the mixing induced by the shortcut left-multiplication, we condition on the branch outputs and treat $\mathbf{A} = \mathbf{I} - \beta\mathbf{k}\mathbf{k}^\top$ as fixed. Consider the special case $d_v = d$ with a diagonal input state $\mathbf{X} \in \mathbb{R}^{d\times d}$, $\mathbf{X} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, which contains no cross-channel coupling. The application of $\mathbf{A}$ yields:

$$(\mathbf{A}\mathbf{X})_{ij} = (\mathbf{X} - \beta\mathbf{k}\mathbf{k}^\top\mathbf{X})_{ij} = s_i\delta_{ij} - \beta s_j k_i k_j,$$

where $\mathbf{X} = \mathrm{diag}(s_1, \ldots, s_d)$ and δ_{ij} is the Kronecker delta. Specifically, the off-diagonal element $(i \neq j)$ becomes $-\beta s_j k_i k_j$, while the diagonal element $(i = j)$ becomes $s_i(1 - \beta k_i^2)$. Thus, even when the incoming state is diagonal, a non-zero β introduces controlled feature coupling proportional to $k_i k_j$. This computation isolates the coupling introduced by the shortcut map $\mathbf{X} \mapsto \mathbf{A}\mathbf{X}$. The full block additionally injects a rank-1 write $\beta\mathbf{k}\mathbf{v}^\top$, i.e., it adds a $\mathbf{k}$-aligned component to each value column.

If $\beta \to 1$, the shortcut removes the component of each column along $\mathbf{k}$, mapping the state into $\mathbf{k}^\perp$ before the write term reinstates a new $\mathbf{k}$-component specified by $\mathbf{v}^\top$. As $\beta \to 0$, both the mixing term and the write vanish, so the diagonal structure is preserved in the limit.

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 Eq. (2.4) simplifies to:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \beta_l \underbrace{(v_l - \mathbf{k}_l^\top\mathbf{x}_l)}_{\gamma_l}\mathbf{k}_l. \tag{3.1}$$

Here, the geometric transformation collapses into a dynamic scalar gating mechanism. The term γ_l acts as a data-dependent coefficient that couples the update magnitude to the discrepancy between the proposed write value v_l and the current projection $\mathbf{k}_l^\top\mathbf{x}_l$.

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 eliminate spatial mixing induced by the shortcut, a sufficient condition is that $\mathbf{k}$ aligns with a canonical basis vector $\mathbf{e}_i$. Then $\mathbf{k}\mathbf{k}^\top = \mathbf{e}_i\mathbf{e}_i^\top$ and $\mathbf{A}$ scales only the i-th coordinate by $(1 - \beta)$ while leaving the other coordinates unchanged. Note, however, that a single rank-1 Delta update writes only along $\mathrm{span}\{\mathbf{k}\}$, recovering a fully general element-wise update across all coordinates would require multiple directions.

3.6 The Delta Rule for Residual Learning

The term 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).

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 Eq. (2.4) using the rank-1 residual definition:

$$\mathbf{X}_{l+1} = \mathbf{X}_l + \beta_l \mathbf{k}_l \Big(\underbrace{\mathbf{v}_l^\top}_{\text{Write target}} - \underbrace{\mathbf{k}_l^\top \mathbf{X}_l}_{\text{Current readout}} \Big),$$

which is exactly Eq. (2.5). This is the matrix-form delta (Widrow-Hoff) update under our left-multiplication convention: the row vector $\mathbf{k}_l^\top \mathbf{X}_l$ contains the current projection coefficients (a readout), and multiplying back by $\mathbf{k}_l$ yields the rank-1 removal term $\beta_l \mathbf{k}_l (\mathbf{k}_l^\top \mathbf{X}_l) = (\beta_l \mathbf{k}_l \mathbf{k}_l^\top) \mathbf{X}_l$. The difference $(\mathbf{v}_l^\top - \mathbf{k}_l^\top \mathbf{X}_l)$ is 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 and injection operations along the geometric direction defined by the projector $\mathbf{k}_l$, modulated by the scalar step size β_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 $\mathbf{k}(\mathbf{X}_l)$. When $\beta(\mathbf{X}_l) \approx 2$, the term subtracts twice the projection, resulting in a sign inversion. This provides the network with a flexible mechanism to selectively clean or reorient specific feature subspaces layer-by-layer, preventing the accumulation of interference. For relations among Deep Delta Learning, Delta Networks, and Householder products, please refer to Appendix A.

4 DDL Transformer

We evaluate Deep Delta Learning as a drop-in replacement for the standard additive residual connection in Transformer language models. Across all comparisons, we keep the backbone architecture fixed (pre-norm RMSNorm, RoPE multi-head attention, and SwiGLU MLP) and change only the residual update rule.

4.1 $d_v = 1$ Transformer setting

In the scalar-value regime ($d_v = 1$), the matrix state degenerates to a vector state $\mathbf{x} \in \mathbb{R}^d$, and the Delta update reduces to Eq. (3.1):

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \beta_l \left(v_l - \mathbf{k}_l^\top \mathbf{x}_l \right) \mathbf{k}_l.$$

We instantiate this update inside each Transformer block. All Transformer experiments use a pre-norm backbone. Let $\mathbf{x}_l^{\text{ctx}} := \text{RMSNorm}(\mathbf{x}_l)$ denote the normalized sublayer input. For a given sublayer (attention or MLP) producing an output $\mathbf{F}(\mathbf{x}_l^{\text{ctx}})$, we compare two parameterization strategies:

- $\mathbf{k}$-Map: We treat the backbone output as the source of the geometric reflection direction $\mathbf{k}$. We set $\widetilde{\mathbf{k}}_l = \mathbf{F}(\mathbf{x}_l^{\text{ctx}})$ and normalize $\mathbf{k}_l = \widetilde{\mathbf{k}}_l / \|\widetilde{\mathbf{k}}_l\|_2$ (with a standard ϵ_k guard in implementation when

$\|\widetilde{\mathbf{k}}_l\|_2$ is extremely small). The scalar update value v_l is derived via a separate linear projection of the residual stream, $v_l = \sigma(\mathbf{w}_v^\top \mathbf{x}_l)$. This formulation interprets the Transformer output as defining *where* to write in the state space, while the content is given by a simple linear projection of the input.

- v-Map: We treat the backbone output as the source of the content value v. We set $v_l = \sigma(\mathbf{w}_p^\top \mathbf{F}(\mathbf{x}_l^{\text{ctx}}))$. The reflection direction is generated by a separate auxiliary branch acting on the context, $\widetilde{\mathbf{k}}_l = \phi_k(\mathbf{x}_l^{\text{ctx}})$, followed by normalization. This formulation interprets the Transformer output as defining *what* to write, decoupling the high-complexity content generation from the geometry, which is determined by a lower-complexity auxiliary map.

In our primary evaluation, we utilize the $\mathbf{k}$-Map variant. In all experiments, the L_2 normalization of $\mathbf{k}_l$ is enforced up to a standard ϵ_k guard when $\|\widetilde{\mathbf{k}}_l\|_2$ is extremely small. In both configurations, the gate β_l is computed via a lightweight linear layer or MLP acting on $\mathbf{x}_l^{\text{ctx}}$.

Precision-friendly normalization. For improved numerical precision in low-precision training, we implement the $\|\mathbf{k}_l\|_2 \approx 1$ constraint via RMS normalization and a fixed scaling factor $k_{\text{scale}} = 1/\sqrt{d}$ rather than explicitly forming $\|\widetilde{\mathbf{k}}_l\|_2$. Our PyTorch implementation computes the unit-L_2 direction using a fused RMSNorm and a constant scale factor:

$$\widehat{\mathbf{k}} = \text{RMSNorm}(\widetilde{\mathbf{k}}; \epsilon_k^2/d), \quad k_{\text{scale}} = \frac{1}{\sqrt{d}}, \quad \mathbf{k} = \widehat{\mathbf{k}}\, k_{\text{scale}}.$$

With $\epsilon_k = 0$, this is exact L_2 normalization. With $\epsilon_k > 0$, it is algebraically equivalent to $\mathbf{k} = \widetilde{\mathbf{k}}/\sqrt{\|\widetilde{\mathbf{k}}\|_2^2 + \epsilon_k^2}$, so $\|\mathbf{k}\|_2 \approx 1$ whenever $\|\widetilde{\mathbf{k}}\|_2 \gg \epsilon_k$, which is the regime assumed by the spectral analysis. This formulation keeps intermediate magnitudes $\mathcal{O}(1)$ and allows applying k_{scale} outside dot products (e.g. in $\mathbf{k}^\top \mathbf{X}$), improving numerical precision in bf16/fp16.

4.2 $d_v > 1$ with State Expansion

In this setting, we relax the constraint that the residual state size must match the backbone compute width. Instead, we investigate a memory-augmented regime where the residual state is explicitly expanded into a matrix $\mathbf{X}_l \in \mathbb{R}^{d \times d_v}$, with $d = D_{\text{model}}$ set to the standard model width and $d_v \geq 2$ acting as a memory expansion factor.

This formulation decouples the memory capacity of the network, which scales linearly with d_v, from the computational capacity of the Transformer backbone, which remains fixed at a width of d. The DDL update follows the standard matrix-valued law:

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

To interface this expanded state with standard Transformer sublayers (Attention and MLP) that expect vector inputs in $\mathbb{R}^d$, we employ a Compress-Process-Expand protocol:

At the embedding layer, we initialize the expanded state by repeating the token embedding across the value channels. Concretely, for an embedding $\mathbf{x}_{\text{emb}} \in \mathbb{R}^d$ we form $\mathbf{X}_0 = \mathbf{x}_{\text{emb}} \mathbf{1}_{d_v}^\top \in \mathbb{R}^{d \times d_v}$ (equivalently, in flattened form, we repeat the length-d embedding to a length-$d \cdot d_v$ vector).

1. **Compression:** We derive the sublayer input $\mathbf{x}_l^{\text{in}} \in \mathbb{R}^d$ by aggregating the memory columns. In our implementation, this readout first applies a short causal depthwise convolution to the expanded stream (independently per channel, over the token dimension) and then performs learned weighted pooling:

$$\widetilde{\mathbf{X}}_l = \text{ShortConv}(\mathbf{X}_l), \qquad \mathbf{x}_l^{\text{in}} = \widetilde{\mathbf{X}}_l \mathbf{w}_p,$$

where $\mathbf{w}_p \in \mathbb{R}^{d_v}$ is a learnable read vector.

2. **Processing:** The standard pre-norm sublayer processes this compressed representation: $\mathbf{h}_l = \mathbf{F}(\text{RMSNorm}(\mathbf{x}_l^{\text{in}})) \in \mathbb{R}^d$.

3. **Expansion:** The backbone output $\mathbf{h}_l$ is used to generate the rank-1 update components. Consistent with the scalar regime ($d_v = 1$), we instantiate the two mapping strategies:

   - **k-Map:** The backbone output determines the reflection direction $\widetilde{\mathbf{k}}_l = \mathbf{h}_l$. The value vector $\mathbf{v}_l \in \mathbb{R}^{d_v}$ is derived via a separate linear projection of the compressed input, $\mathbf{v}_l = \mathbf{W}_v \mathbf{x}_l^{\text{in}}$, where $\mathbf{W}_v \in \mathbb{R}^{d_v \times d}$.
   - **v-Map**: The backbone output determines the value content $\mathbf{v}_l = \mathbf{W}_v \mathbf{h}_l$, where $\mathbf{W}_v \in \mathbb{R}^{d_v \times d}$. The reflection geometry $\mathbf{k}_l$ is derived from the sublayer input context via an auxiliary branch, $\widetilde{\mathbf{k}}_l = \phi_k(\mathbf{x}_l^{\text{in}})$.

This architecture interprets the hidden state as a short-term memory bank of d_v slots per feature. The Delta Operator synchronizes the update across these slots, enforcing a shared geometric transformation while allowing the value content to vary across the memory dimension. We evaluate this setting with $d_v = 4$, representing a $4\times$ expansion of the residual memory footprint relative to the baseline Transformer, without increasing the FLOPs of the attention mechanism.

5 Experiments

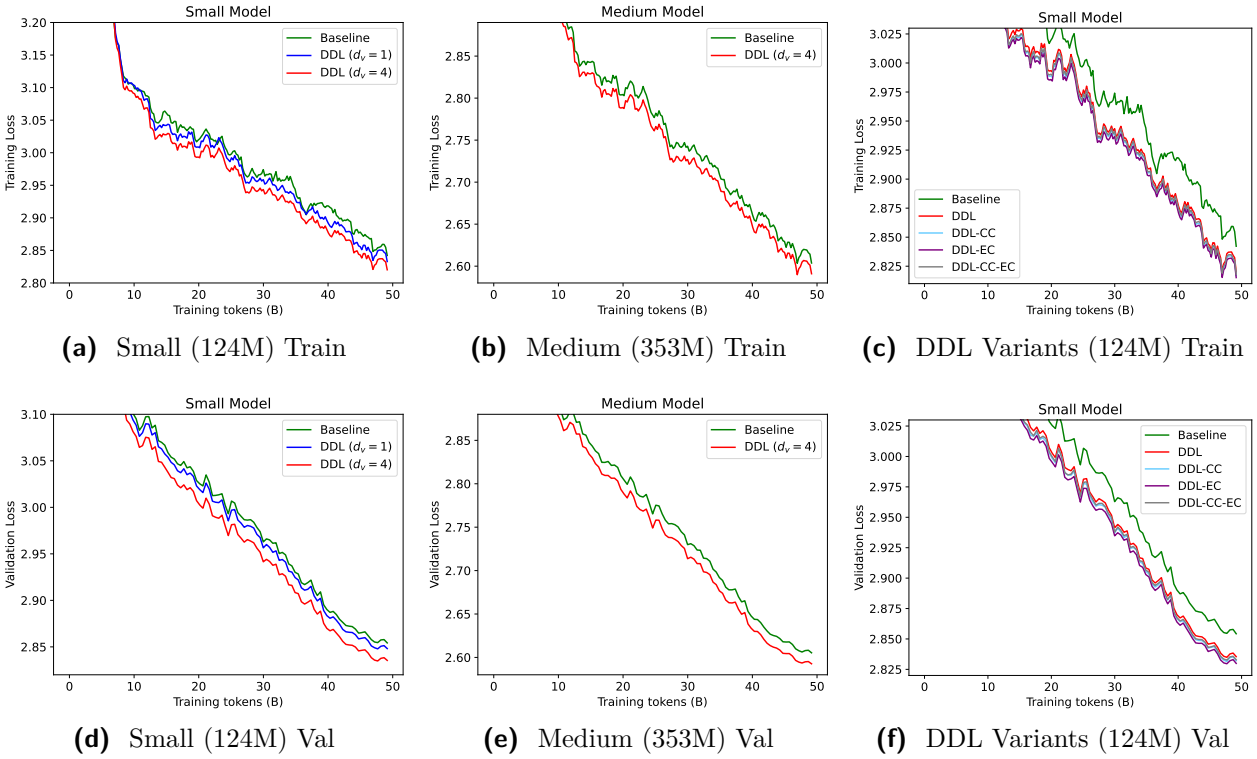
We compare our proposed Deep Delta Learning (DDL) with $d_v = 1$ and $d_v = 4$ against a baseline based on the nanoGPT codebase (Karpathy, 2022). For the DDL model with $d_v = 4$, we use depthwise convolution (Chollet, 2017) along the sequence-length dimension with $d \cdot d_v$ channels.

Table 1 The evaluation results of small-size models on the FineWeb-Edu 100B dataset (1-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	29.01	55.85	37.59	30.20	65.94	80.60	37.87	51.38	48.56
DDL ($d_v = 1$)	**29.35**	57.49	38.08	31.80	64.85	78.50	37.77	52.01	48.73
DDL ($d_v = 4$)	27.90	58.16	38.26	30.80	**66.49**	80.30	**38.54**	50.83	48.91
DDL-CC	27.82	**58.92**	**38.44**	33.20	65.83	79.50	38.28	51.07	49.13
DDL-EC	28.75	57.37	38.41	**34.40**	64.47	82.00	38.38	52.01	**49.47**
DDL-CC-EC	28.33	57.87	38.24	32.20	64.09	**82.60**	38.43	**52.57**	49.29

Table 2 The evaluation results of medium-size models on the FineWeb-Edu 100B dataset (1-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	33.62	**67.05**	47.42	33.20	70.24	87.30	40.28	52.57	53.96
DDL ($d_v = 1$)	**35.49**	65.70	46.94	34.20	**70.35**	88.90	**40.99**	54.93	54.69
DDL ($d_v = 4$)	33.02	66.16	**47.83**	**35.60**	69.86	**89.50**	**40.99**	**55.64**	**54.83**



(a) Small (124M) Train **(b)** Medium (353M) Train **(c)** DDL Variants (124M) Train

(d) Small (124M) Val **(e)** Medium (353M) Val **(f)** DDL Variants (124M) Val

Figure 2 Loss curves on FineWeb-Edu 100B. The top row displays training loss; bottom row displays validation loss. Columns represent the Small (124M), Medium (353M), and Small DDL Variant architectures, respectively.

Table 3 The valid loss and perplexity of small-size and medium-size models at the final step on the FineWeb-Edu 100B dataset. The best losses and perplexities in each column are bolded.

Model	Small		Medium	
	Valid Loss	Valid Perplexity	Valid Loss	Valid Perplexity
Baseline	2.85426	17.3616	2.60532	13.5356
DDL ($d_v = 1$)	2.84817	17.2562	2.60388	13.5161
DDL ($d_v = 4$)	<u>2.83545</u>	<u>17.0381</u>	**2.59267**	**13.3654**
DDL-EC ($d_v = 4$)	**2.82990**	**16.9438**		

5.1 Experimental settings

We train on FineWeb-Edu 100B (Lozhkov et al., 2024), which contains 100 billion training tokens and 0.1 billion validation tokens. We run experiments at two scales: small (124M parameters) and medium (353M parameters), using Llama-style models with RoPE (Su et al., 2024) embeddings and SwiGLU activations (Shazeer, 2020); we also add query and key normalization to stabilize training.

We train our models for 100,000 steps with a global batch size of 480 and a sequence length of 1,024. Additionally, we use μP initialization and parameterization (Yang et al., 2022). The learning rate is set to 1e$-$3 with a cosine learning-rate schedule and 2,000 warmup steps. We use AdamW (Loshchilov and Hutter, 2019) with weight decay 0.1, $(\beta_1, \beta_2) = (0.9, 0.95)$, and gradient clipping at 1.0. We disable bias terms and set the dropout rate to 0.0. For each experiment, we use

4 NVIDIA H200 GPUs. Other hyperparameters are listed in Appendix C.

5.2 Experimental Results

The training and validation loss curves are shown in Figures 2a, 2b, and Figures 2d, 2e, and the losses and perplexities at the final step are reported in Table 3. The results show that DDL with $d_v = 4$ outperforms the baseline models, and that DDL with $d_v = 1$ also achieves lower training and validation loss than the baseline.

Moreover, we evaluate 1-shot and 0-shot performance on ARC (Yadav et al., 2019), HellaSwag (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), Social IQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021) using the `lm-evaluation-harness` codebase (Gao et al., 2021). The 1-shot results are shown in Tables 1 and 2, and 0-shot results are postponed to Appendix D. Overall, DDL performs better than the baseline, with further improvements as d_v increases.

5.3 Ablation Studies

Moreover, we also implement some variants of DDL [1] with $d_v = 4$, including the model with channel convolution along the d_v dimension (denoted as "DDL-CC") and their variants with a convolution after input embedding along the sequence length dimension (denoted as "DDL-EC" and "DDL-CC-EC", respectively). The training and validation loss curves are shown in Figures 2c and 2f. And the 1-shot and 0-shot evaluation results on benchmarks are displayed in Table 1 and 5 (in Appendix D). The results show that adding embedding convolution and convolution along d_v may further improve the performance of the models.

6 Related Work

Our work builds on several key research themes in deep learning, including gated and invertible architectures, orthogonal and unitary networks, and neural ordinary differential equations.

Gated and Invertible Architectures. Highway Networks (Srivastava et al., 2015) introduced data-dependent gating to residual networks, and later, Chai et al. (2020) augmented Highway Networks by introducing multi-layer feed-forward networks to the gate. Menghani et al. (2025), Pagliardini et al. (2024), and Fang et al. (2023) further introduced dense connections from previous layers to facilitate information flow, and Xiao et al. (2025) uses an additional residual way with dynamic dense connections for cross-layer information transfer. However, their gates interpolate between the identity path and the function path, rather than modifying the transformation itself. Mak and Flanigan (2025) introduced outer-product memory stores into Transformers for information storage and retrieval. Recently, Zhu et al. (2025) proposed Hyper-connection, which introduced depth and width connections to integrate features across variable depths. Later, Xie et al. (2025) improves upon this by introducing manifold projection to maintain the identity mapping property. 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 shortcut operator is invertible whenever $1-\beta \neq 0$ (equivalently, $\beta \neq 1$), and becomes an orthogonal involution at $\beta = 2$ (a Householder reflection). DDL does not enforce invertibility globally; instead, it allows

[1] Detailed introduction of these variants is presented in Appendix B.3.

the network to learn when a near-invertible transition is beneficial versus when an intentionally singular (projective) transition is useful for controlled forgetting.

Orthogonal and Unitary Networks. Many previous works have focused on constraining network weights to be orthogonal (Zhang et al., 2021; Wang et al., 2025) or unitary (Arjovsky et al., 2016; Jing et al., 2017; He et al., 2025) to improve gradient stability and preserve geometric structure. Fei et al. (2022) orthogonalize the self-attention parameters in the vision Transformer to retain the geometry of the feature space. Xie et al. (2025) introduced mHC, which uses the Sinkhorn–Knopp operation to rescale the residual matrix to have orthogonal properties. Householder reflections are a classic method for parameterizing orthogonal matrices. Yang et al. (2025) utilized this property in position encoding, while Dong et al. (2024) applied it in vision Transformers to efficiently mimic singular value decomposition. Arcas et al. (2025) integrate it into low-rank adaptation to reduce time and space complexity. 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 reflection.

Neural Ordinary Differential Equations. Neural Ordinary Differential Equations (ODEs Chen et al. (2018)) model the continuous evolution of features. Prior work has integrated ODE models into Transformers (Tong et al., 2025; Li et al., 2022; Kan et al., 2025; Jemley, 2026) to facilitate time-series modeling (Chen et al., 2023; Zhang et al., 2025) and to enable depth-adaptive Transformers (Baier-Reinio and De Sterck, 2020). ODE-RNN (Rubanova et al., 2019) combines Neural ODEs with gated recurrent units (Cho et al., 2014) for irregular time-series modeling. For Deep Delta Network, the standard ResNet Eq. (1.1) is a discretization of the ODE $\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X})$, while our update $\mathbf{X}_{l+1} - \mathbf{X}_l = \beta(\mathbf{X}_l)\,\mathbf{k}(\mathbf{X}_l)(\mathbf{v}(\mathbf{X}_l)^\top - \mathbf{k}(\mathbf{X}_l)^\top \mathbf{X}_l)$ can be viewed as a forward Euler step for $\dot{\mathbf{X}} = \mathbf{k}(\mathbf{X})(\mathbf{v}(\mathbf{X})^\top - \mathbf{k}(\mathbf{X})^\top \mathbf{X})$ with an adaptive (state-dependent) step size $\beta(\mathbf{X})$, yielding a family of depth-wise dynamics that can be contractive or sign-flipping along a learned direction.

7 Conclusion

We have introduced Deep Delta Learning, a model 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 principled method for increasing expressive power while retaining the foundational benefits of the residual learning paradigm.

References

Alejandro Moreno Arcas, Albert Sanchis, Jorge Civera, and Alfons Juan. Hoft: Householder orthogonal fine-tuning. *arXiv preprint arXiv:2505.16531*, 2025.

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

Aaron Baier-Reinio and Hans De Sterck. N-ode transformer: A depth-adaptive variant of the transformer using neural ordinary differential equations. *arXiv preprint arXiv:2010.11358*, 2020.

Cenk Baykal, Dylan Cutler, Nishanth Dikkala, Nikhil Ghosh, Rina Panigrahy, and Xin Wang. Alternating updates for efficient transformers. *Advances in Neural Information Processing Systems*, 36:76718–76736, 2023.

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.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

Yekun Chai, Shuo Jin, and Xinwen Hou. Highway transformer: Self-gating enhanced self-attentive networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6887–6900, 2020.

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

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36:47143–47175, 2023.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Wei Dong, Yuan Sun, Yiting Yang, Xing Zhang, Zhijun Lin, Qingsen Yan, Haokui Zhang, Peng Wang, Yang Yang, and Hengtao Shen. Efficient adaptation of pre-trained vision transformer via householder transformation. *Advances in Neural Information Processing Systems*, 37:102056–102077, 2024.

Yanwen Fang, CAI Yuxi, Jintai Chen, Jingyu Zhao, Guangjian Tian, and Guodong Li. Cross-layer retrospective retrieving via layer attention. In *The Eleventh International Conference on Learning Representations*, 2023.

Yanhong Fei, Yingjie Liu, Xian Wei, and Mingsong Chen. O-vit: Orthogonal vision transformer. *arXiv preprint arXiv:2201.12133*, 2022.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025.

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.

Yi He, Yiming Yang, Xiaoyuan Cheng, Hai Wang, Xiao Xue, Boli Chen, and Yukun Hu. Chaos meets attention: Transformers for large-scale dynamical prediction. *arXiv preprint arXiv:2504.20858*, 2025.

Alston S Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4):339–342, 1958.

Peter Jemley. Continuous-depth transformers with learned control dynamics. *arXiv preprint arXiv:2601.10007*, 2026.

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.

Kelvin Kan, Xingjian Li, and Stanley Osher. Ot-transformer: a continuous-time transformer architecture with optimal transport regularization. *arXiv preprint arXiv:2501.18793*, 2025.

Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022.

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8335–8351, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.571. URL https://aclanthology.org/2022.acl-long.571/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024.

Brian Mak and Jeffrey Flanigan. Residual matrix transformers: Scaling the size of the residual stream. In *Forty-second International Conference on Machine Learning*, 2025.

Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. Laurel: Learned augmented residual layer. In *Forty-second International Conference on Machine Learning*, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

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.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Anh Tong, Thanh Nguyen-Tang, Dongeun Lee, Duc Nguyen, Toan Tran, David Leo Wright Hall, Cheongwoong Kang, and Jaesik Choi. Neural ode transformers: Analyzing internal dynamics and adaptive fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xiangming Wang, Haijin Zeng, Jiaoyang Chen, Sheng Liu, Yongyong Chen, and Guoqing Chao. Otlrm: Orthogonal learning-based low-rank metric for multi-dimensional inverse problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21278–21286, 2025.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. In *Forty-second International Conference on Machine Learning*, 2025.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Liang Zhao, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. *arXiv preprint arXiv:1911.07176*, 2019.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. Path attention: Position encoding via accumulating householder transformations. *arXiv preprint arXiv:2505.16381*, 2025.

Aston Zhang, Alvin Chan, Yi Tay, Jie Fu, Shuohang Wang, Shuai Zhang, Huajie Shao, Shuochao Yao, and Roy Ka-Wei Lee. On orthogonality constraints for transformers. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 375–382, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.48. URL https://aclanthology.org/2021.acl-short.48/.

Yudong Zhang, Xu Wang, Xuan Yu, Zhengyang Zhou, Xing Xu, Lei Bai, and Yang Wang. Diffode: Neural ode with differentiable hidden state for irregular time series analysis. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 1–14. IEEE, 2025.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *The Thirteenth International Conference on Learning Representations*, 2025.

Appendix

A 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 $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$:

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

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 Eq. (2.4) acting over depth l: Eq. (A.1) is algebraically equivalent to the more common right-multiplication delta-rule update $\mathbf{M}_t = \mathbf{M}_{t-1} + \beta_t(\mathbf{v}_t - \mathbf{M}_{t-1}\mathbf{k}_t)\mathbf{k}_t^\top$ by setting $\mathbf{S}_t = \mathbf{M}_t^\top$ (a transpose convention for the memory matrix).

$$\mathbf{X}_{l+1} = (\mathbf{I} - \beta_l \mathbf{k}_l \mathbf{k}_l^\top)\mathbf{X}_l + \beta_l \mathbf{k}_l \mathbf{v}_l^\top, \tag{A.2}$$

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

This reveals a direct structural correspondence:

- 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.
- Both architectures employ the same rank-1 shortcut operator $\mathbf{I} - \beta\,\mathbf{k}\mathbf{k}^\top$: under $\|\mathbf{k}\|_2 = 1$ it is an orthogonal projector at $\beta = 1$ and a Householder reflection at $\beta = 2$. DeltaNet applies it over time steps t, whereas DDL applies it over network depth l.
- Our modified residual update $\beta_l \mathbf{k}_l \mathbf{v}_l^\top$ aligns perfectly with the DeltaNet write operation. By incorporating β_l into the constructive term, we interpret β_l as a layer-wise step size for the depth-wise ODE. This ensures that both the erasure and injection 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.

B 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 $\phi_k : \mathbb{R}^{d \times d_v} \to \mathbb{R}^d$.

B.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 ϕ_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.

$$\widetilde{\mathbf{k}}_{\text{MLP}} = \text{MLP}(\text{Pool}(\mathbf{X}))\,, \quad \mathbf{k}_{\text{MLP}} = \frac{\widetilde{\mathbf{k}}_{\text{MLP}}}{\|\widetilde{\mathbf{k}}_{\text{MLP}}\|_2}$$

Here, $\text{Pool}(\cdot)$ is any aggregation that produces a fixed-size vector representation of $\mathbf{X}$, e.g., column-wise averaging ($\mathbb{R}^{d\times d_v} \to \mathbb{R}^d$) or flattening ($\mathbb{R}^{d\times d_v} \to \mathbb{R}^{d\cdot d_v}$), followed by an MLP that outputs $\mathbb{R}^d$. We enforce L_2 normalization to satisfy the spectral assumptions in Theorem 3.1. In implementation, the division is guarded by a small ϵ_k only when $\|\widetilde{\mathbf{k}}_{\text{MLP}}\|_2$ is extremely small; the analysis assumes $\|\mathbf{k}\|_2 = 1$.

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

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

The Gating Branch. The scalar gate β requires a bounded output in $[0, 2]$. In `model/DDL-gpt-mha-rope*.py`, β is computed per token from the same pre-norm context used by the backbone, $\mathbf{c} = \text{RMSNorm}(\mathbf{x}^{\text{in}}) \in \mathbb{R}^d$ (where $\mathbf{x}^{\text{in}}$ is the compressed readout of the expanded state described above). We keep this estimator lightweight and configurable (`ddl_beta_single_linear`, `ddl_beta_hidden_size`):

$$\beta(\mathbf{c}) = 2 \cdot \sigma(\text{Linear}(\mathbf{c})) \quad \text{or} \quad \beta(\mathbf{c}) = 2 \cdot \sigma\big(\text{Linear}\big(\tanh(\text{Linear}(\mathbf{c}))\big)\big)\,.$$

We compute the logits in fp32 for stability and initialize the output bias to match a desired starting value $\beta_0 \in [0, 2]$ via $\text{logit}(\beta_0/2)$ (clamped in code; `ddl_beta_init`), 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, $\phi_v : \mathbb{R}^{d\times d_v} \to \mathbb{R}^{d_v}$, 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, ϕ_v mirrors the Feed-Forward Network or Multi-Head Attention block structure) to ensure capacity alignment.

B.3 Expanded-state Transformer Implementation Details

Our PyTorch implementations for the expanded-state Transformer variant ($d_v > 1$) live in `model/DDL-gpt-mha-rope*.py`. For clarity, we summarize the common tensor layout and then highlight the key differences between the repository variants.

Repository variants.

- `model/DDL-gpt-mha-rope.py`: baseline expanded-state DDL on top of GPT (MHA + RoPE), initializing the state by repeating token embeddings across the value-channel axis and compressing the expanded residual with a token-axis short causal convolution plus a learned read vector.
- `model/DDL-gpt-mha-rope-EC.py`: same as the baseline, but expands token embeddings with a learnable depthwise causal short convolution (`input_embed_shortconv_kernel_size`, identity-initialized).
- `model/DDL-gpt-mha-rope-CC.py`: same as the baseline, but switches the residual-compression short convolution to operate along the value-channel axis d_v and consume it (Conv returns length 1; `ddl_state_shortconv_kernel_size` $= d_v$).

- `model/DDL-gpt-mha-rope-CC-EC.py`: same as `model/DDL-gpt-mha-rope-CC.py`, but expands token embeddings with a learnable depthwise causal short convolution (`input_embed_shortconv_kernel_size`, identity-initialized).

State layout and initialization. For a batch of sequences, we represent the expanded residual as a rank-4 tensor of shape (B, T, d, d_v), where d is the model width and d_v is the number of value channels.

Input expansion. In the baseline and CC modules, we initialize $\mathbf{X}_0 = \mathbf{x}_{\text{emb}} \mathbf{1}_{d_v}^\top$ by repeating the embedding across the value axis (implemented as `x_emb.unsqueeze(-1).repeat(..., d_v)`). In the EC module, we instead use a depthwise causal short convolution over the token dimension to map $(B, T, d) \to (B, T, d \cdot d_v)$ and then reshape to (B, T, d, d_v); the convolution is identity-initialized, so the starting behavior matches simple repetition.

ShortConv compression along T (baseline, DDL-EC). In `model/DDL-gpt-mha-rope.py` and `model/DDL-gpt-mha-rope-EC.py`, the `ResidualShortConvCompressor` first flattens the last two dimensions, treating the expanded residual as $(B, T, d \cdot d_v)$ channels, applies a short causal depthwise Conv1d over the token dimension (kernel size `ddl_state_shortconv_kernel_size`), reshapes back to (B, T, d, d_v), and then pools across value channels with a learned read vector $\mathbf{w}_p \in \mathbb{R}^{d_v}$:

$$\widetilde{\mathbf{X}}_{l,t,i,j} = \sum_{s=0}^{k-1} c_{i,j,s} \, \mathbf{X}_{l,t-s,i,j}, \qquad \mathbf{x}^{\text{in}}_{l,t,i} = \sum_{j=1}^{d_v} w_{p,j} \, \widetilde{\mathbf{X}}_{l,t,i,j}.$$

We initialize the read vector to a uniform average by default (`ddl_state_read_init` $= 1/d_v$ when unset).

ShortConv compression along d_v (DDL-CC, DDL-CC-EC). In `model/DDL-gpt-mha-rope-CC.py` and `model/DDL-gpt-mha-rope-CC-EC.py`, we instead apply a depthwise causal Conv1d along the value-channel axis and consume d_v in one shot (the Conv returns length 1). Concretely, we reshape the state to $(B \cdot T, d, d_v)$ and treat d_v as the convolution "sequence" length, using per-feature kernels $c_{i,j}$ with kernel size $k = d_v$:

$$\mathbf{x}^{\text{in}}_{l,t,i} = \sum_{j=1}^{d_v} c_{i,j} \, \mathbf{X}_{l,t,i,j}.$$

This changes the locality prior from time-local mixing to value-channel-local mixing. Note that in CC, `ddl_state_shortconv_kernel_size` refers to the kernel size along d_v (not along T), and must equal d_v for the Conv to return length 1.

C Hyper-parameter Settings

Here we just list the hyperparameters for small and medium sizes of models here in Table 4.

Table 4 Architecture hyper-parameters for small and medium sizes of models.

Model	#Param	#Layer	#Head	Head Dimension	Hidden Size
Small-size Model	124M	12	6	128	768
Medium-size Model	353M	24	8	128	1024

D Additional Results

We just display the 0-shot evaluation results of small-size and medium size models in Tables 5 and 6, respectively. The results also demonstrate that DDL can achieve better performances on downstream tasks.

Table 5 The evaluation results of small-size models on the FineWeb-Edu 100B dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	28.33	52.44	37.60	33.00	65.94	71.20	37.46	52.41	47.30
DDL ($d_v = 1$)	26.96	52.40	37.91	32.20	64.91	72.50	38.08	**53.59**	47.32
DDL ($d_v = 4$)	27.30	**52.95**	38.40	**33.80**	65.40	**73.70**	38.64	50.12	47.54
DDL-CC	27.05	51.47	38.72	32.20	**66.16**	71.80	38.08	51.07	47.07
DDL-EC	**28.50**	51.89	**38.82**	33.20	65.29	73.00	**39.05**	52.88	**47.83**
DDL-CC-EC	27.90	51.26	38.42	31.40	64.85	73.60	37.77	52.25	47.18

Table 6 The evaluation results of medium-size models on the FineWeb-Edu 100B dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	31.74	**59.85**	47.91	34.00	69.21	78.10	**40.69**	53.83	51.92
DDL ($d_v = 1$)	32.07	59.39	47.62	34.40	**70.08**	77.30	39.61	**55.01**	51.94
DDL ($d_v = 4$)	**32.08**	58.38	**48.08**	**35.80**	69.42	**79.90**	39.92	54.14	**52.22**