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

Abstract

Transformer residual streams evolve through additive updates. Although a sufficiently expressive residual block can represent content replacement, standard architectures do not parameterize reading, comparison, and replacement as an explicit residual operation. We introduce **Deep Delta Learning (DDL)**, a structured residual update that preserves the identity path while enabling target-seeking edits to the residual state. Each layer reads the current state along a learned direction, compares the resulting readout with a learned target, and writes back a gated rank-1 correction along the same direction. Closing the gate recovers the identity map, while fully opening it exactly overwrites the selected residual readout. We instantiate DDL with both scalar and expanded residual states. The expanded formulation provides multiple persistent value channels while keeping attention and MLP computation at the original model width, thereby separating residual-state capacity from backbone compute width. Controlled LLM pretraining experiments show that DDL improves language-modeling quality and average one-shot downstream performance over additive residual baselines in the reported runs, while introducing explicit memory and throughput tradeoffs. These results suggest that depth-wise delta-rule updates provide a useful inductive bias for managing Transformer residual streams.

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

1 Introduction

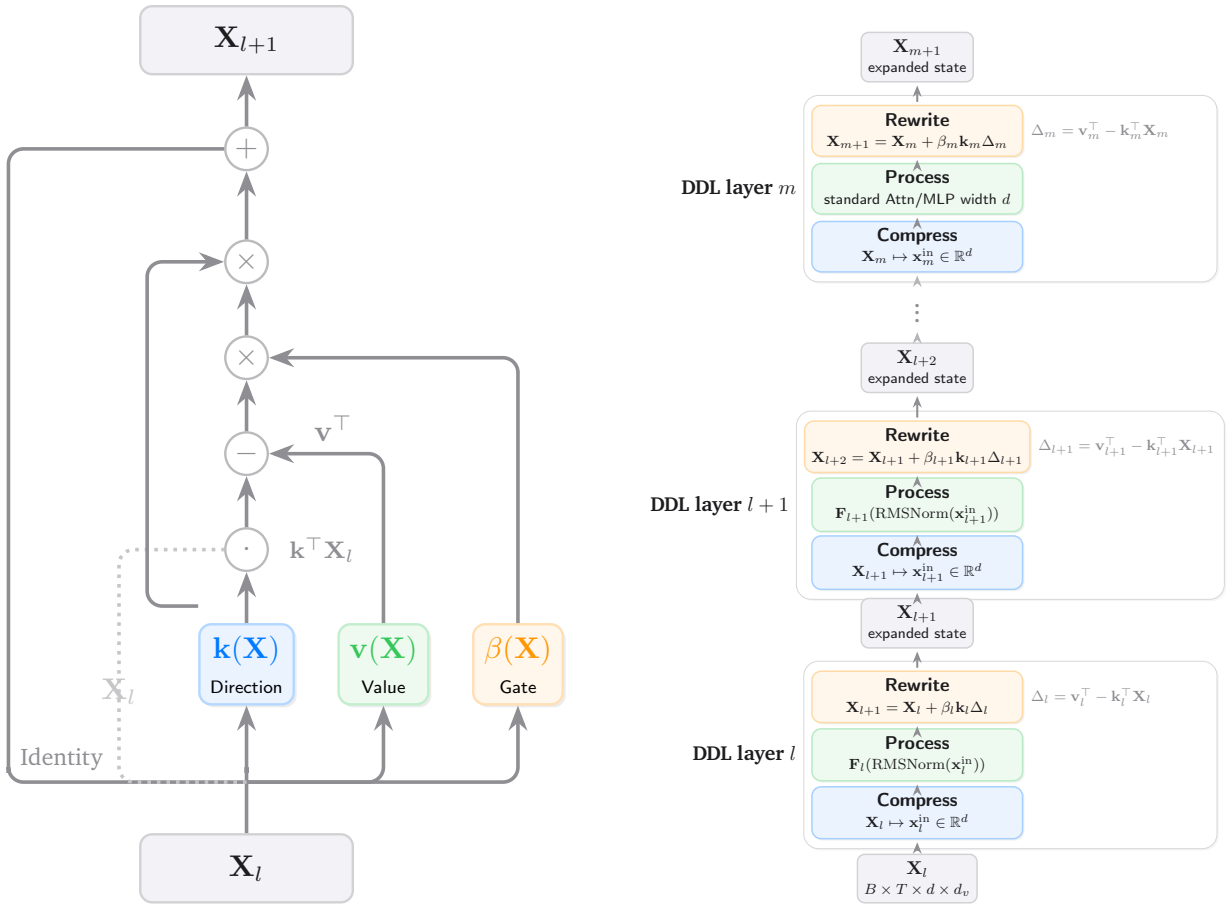
Residual connections provide the default interface for composing very deep networks: an identity path stabilizes optimization while each block contributes an incremental transformation (He et al., 2016). In Transformer language models, the same interface is also the persistent token-level state shared by all attention and MLP blocks. A standard block updates this state by

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathbf{F}_l(\mathbf{x}_l),$$

where $\mathbf{x}_l \in \mathbb{R}^{B\times T\times d}$ for batch size B, sequence length T, and compute width d. This rule is simple, stable, and hardware-friendly. It can also represent replacement in principle: a sufficiently expressive branch may learn an update that cancels a current value and inserts a new one. The distinction we study is therefore not one of strict expressivity. Rather, standard residual addition leaves reading, comparison, cancellation, and writing implicit inside an unconstrained vector-valued branch.

We introduce **Deep Delta Learning (DDL)**, a residual interface that parameterizes these operations explicitly through a depth-wise delta rule:

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



(a) Deep Delta residual block. The architecture generalizes the standard residual connection. A learnable scalar gate β controls a shortcut operator with a rank-1 perturbation.

(b) DDL Transformer structure. A DDL Transformer repeatedly applies a Compress-Process-Rewrite interface across depth.

Figure 1 Deep Delta Learning overview. (a) The Deep Delta residual block generalizes the standard residual connection with a gated rank-1 rewrite operation. (b) A DDL Transformer repeatedly applies a Compress-Process-Rewrite interface across depth, where the Process part generates the DDL key $\mathbf{k}$ and value $\mathbf{v}$ used by the Rewrite update, while keeping attention and MLP sublayers at width d. Each layer compresses the expanded residual state to a width-d token representation, processes it with the standard attention or MLP sublayer, uses the Process output and normalized residual context to generate the DDL key $\mathbf{k}_l$, value $\mathbf{v}_l$, and gate β_l, and then rewrites the persistent expanded state through the DDL read-compare-write update.

The update reads the current residual state along a learned unit direction $\mathbf{k}_l$, compares the readout with a learned target $\mathbf{v}_l^\top$, and writes the gated discrepancy back along the same direction. A shared scalar gate β_l synchronizes the erase and write terms. Closing the gate recovers the identity map; setting $\beta_l = 1$ exactly matches the selected readout to the target. DDL remains additive in the broad algebraic sense that it has the form $\mathbf{X}_{l+1} = \mathbf{X}_l + \Delta\mathbf{X}_l$. Its contribution is a structured, target-seeking rank-1 parameterization of $\Delta\mathbf{X}_l$, not a claim that ordinary residual blocks are incapable of representing the same function.

This parameterization gives a direct local error-correction interpretation. Conditioned on the

generated $\mathbf{k}_l$, $\mathbf{v}_l$, and β_l, define

$$\mathbf{e}_l^{\mathrm{pre}} = \mathbf{k}_l^\top \mathbf{X}_l - \mathbf{v}_l^\top, \qquad \mathbf{e}_l^{\mathrm{post}} = \mathbf{k}_l^\top \mathbf{X}_{l+1} - \mathbf{v}_l^\top.$$

Then $\mathbf{e}_l^{\mathrm{post}} = (1-\beta_l)\mathbf{e}_l^{\mathrm{pre}}$. Thus $0 < \beta_l < 2$ contracts the selected-coordinate error in magnitude, $\beta_l = 1$ removes it exactly, and $\beta_l > 1$ applies an over-relaxed correction. A conventional residual branch can learn this map, but DDL makes the target, discrepancy, edit direction, and step size explicit architectural quantities. The underlying delta rule is established in sequence-memory models (Schlag et al., 2021; Yang et al., 2024); our contribution is to use it as a residual interface over network depth and to analyze the resulting local shortcut geometry.

DDL can operate on the ordinary scalar residual state ($d_v = 1$) or on an expanded residual state $\mathbf{X}_l \in \mathbb{R}^{B\times T\times d\times d_v}$ with a small number of value channels. For $d_v > 1$, a learned compressor maps the persistent state to width d before the standard attention or MLP sublayer. The expanded state therefore increases persistent residual capacity without widening attention keys, queries, values, or MLP activations. It also introduces additional memory, bandwidth, and compression cost, and its empirical gains must be interpreted jointly with those architectural changes.

We evaluate DDL in decoder-only language models trained on FineWeb-Edu for 49.15B tokens at approximately GPT-2 small and GPT-2 medium parameter budgets. In the scalar setting, which does not expand residual capacity, the single-run point estimates reduce validation loss by 0.0061 and 0.0014 and raise one-shot average accuracy by 0.17 and 0.73 points at the two scales. The best expanded-state variants reduce validation loss by 0.0244 and 0.0295 and raise one-shot averages by 0.91 and 1.18 points, but these models also change residual capacity, input expansion, and compression. The experiments are equal-token rather than iso-FLOPs comparisons and contain one run per configuration; we therefore present them as evidence of architectural promise and measured quality–cost tradeoffs, not as a statistically resolved attribution of all gains to the erase term.

Our contributions are:

1. We formulate **Deep Delta Learning**, a depth-wise residual interface whose correction is a shared-gate, rank-1 read-compare-write update. DDL preserves the identity limit and exactly matches a learned target along the selected one-dimensional readout when $\beta_l = 1$.
2. We characterize the conditioned shortcut operator $\mathbf{I} - \beta_l \mathbf{k}_l \mathbf{k}_l^\top$ and the induced selected-coordinate error update. The analysis separates identity-like, exact-overwrite, and over-relaxed regimes and provides operator-level semantics without claiming semantic interpretability of the learned direction.
3. We instantiate DDL with scalar and expanded residual states while keeping attention and MLP computation at width d. Equal-token pretraining, downstream evaluation, throughput, and memory measurements at two model scales quantify the observed benefits and costs, while making explicit the unresolved need for multi-seed, matched expanded-state controls, and iso-FLOPs comparisons.

2 Deep Delta Learning

Deep Delta Learning changes the parameterization of the residual update, not the function class of the attention or MLP sublayer. It is algebraically additive:

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

A sufficiently expressive conventional residual branch could represent the same $\Delta\mathbf{X}_l$. DDL instead constrains the correction to a target-seeking rank-1 form whose read, target, direction, and step size are explicit. We derive the update for one token and one layer, suppressing batch and sequence axes. The residual state is $\mathbf{X}_l \in \mathbb{R}^{d\times d_v}$, where d is the Transformer width and d_v is the number of residual value channels; $d_v = 1$ recovers an ordinary vector state.

2.1 Delta Residual Rewrite

Given $\mathbf{X}_l$, three generator functions produce an unnormalized direction, a target value, and a gate:

$$\widetilde{\mathbf{k}}_l = \mathcal{K}_l(\mathbf{X}_l) \in \mathbb{R}^d, \qquad \mathbf{v}_l = \mathcal{V}_l(\mathbf{X}_l) \in \mathbb{R}^{d_v}, \qquad \beta_l = \mathcal{B}_l(\mathbf{X}_l) \in (0, 2).$$

Here $\mathcal{K}_l$, $\mathcal{V}_l$, and $\mathcal{B}_l$ denote functions, whereas $\widetilde{\mathbf{k}}_l$, $\mathbf{v}_l$, and β_l denote their outputs. Section 3 specifies the concrete Transformer parameterization used in the experiments: $\mathcal{K}_l$ contains the standard sublayer, while $\mathcal{V}_l$ and $\mathcal{B}_l$ are lightweight projections from the normalized residual context.

Read/write direction. We normalize the proposed direction as

$$\mathbf{k}_l = \frac{\widetilde{\mathbf{k}}_l}{\|\widetilde{\mathbf{k}}_l\|_2}, \qquad \|\mathbf{k}_l\|_2 = 1,$$

with a small-norm guard in implementation. The row vector $\mathbf{k}_l^\top \mathbf{X}_l \in \mathbb{R}^{1\times d_v}$ is the current readout along the selected feature direction, and $\mathbf{v}_l^\top$ is its target value.

Erase and write as one affine update. Conditioned on the generated quantities, define the direct shortcut operator

$$\mathbf{A}_l = \mathbf{I} - \beta_l \mathbf{k}_l \mathbf{k}_l^\top.$$

DDL applies this shortcut and writes a rank-1 target along the same direction:

$$\mathbf{X}_{l+1} = \mathbf{A}_l \mathbf{X}_l + \beta_l \mathbf{k}_l \mathbf{v}_l^\top. \tag{2.1}$$

Equivalently,

$$\mathbf{X}_{l+1} = \mathbf{X}_l + \beta_l \mathbf{k}_l \Big(\mathbf{v}_l^\top - \mathbf{k}_l^\top \mathbf{X}_l\Big). \tag{2.2}$$

The discrepancy $\mathbf{v}_l^\top - \mathbf{k}_l^\top \mathbf{X}_l \in \mathbb{R}^{1\times d_v}$ vanishes when the selected readout already matches the target. Multiplication by $\mathbf{k}_l$ confines the direct correction to $\mathrm{span}\{\mathbf{k}_l\}$; directions orthogonal to $\mathbf{k}_l$ remain on the identity shortcut after conditioning on the generated quantities.

Shared gate and exact overwrite. The same β_l controls both erasure and writing. At $\beta_l = 0$, Eq. (2.2) gives $\mathbf{X}_{l+1} = \mathbf{X}_l$. At $\beta_l = 1$,

$$\mathbf{k}_l^\top \mathbf{X}_{l+1} = \mathbf{v}_l^\top, \tag{2.3}$$

so the one-dimensional readout along $\mathbf{k}_l$ is exactly replaced by the target. Separate erase and write gates would define a more general affine update, but would no longer enforce this synchronized target-matching interpretation.

Target-seeking error correction. Let

$$\mathbf{e}_l^{\mathrm{pre}} = \mathbf{k}_l^\top \mathbf{X}_l - \mathbf{v}_l^\top, \qquad \mathbf{e}_l^{\mathrm{post}} = \mathbf{k}_l^\top \mathbf{X}_{l+1} - \mathbf{v}_l^\top.$$

Using $\|\mathbf{k}_l\|_2 = 1$ in Eq. (2.2) gives

$$\mathbf{e}_l^{\text{post}} = (1 - \beta_l)\mathbf{e}_l^{\text{pre}}. \tag{2.4}$$

Therefore $0 < \beta_l < 2$ contracts the discrepancy in magnitude by $|1 - \beta_l|$, $\beta_l = 1$ removes it exactly, and $1 < \beta_l < 2$ changes its sign while reducing its magnitude. This statement is local and conditioned on the generated $\mathbf{k}_l$, $\mathbf{v}_l$, and β_l; it is not a claim that the full nonlinear layer or the end-to-end network is globally contractive.

Gate parameterization. In the reported models,

$$\beta_l = 2 \cdot \sigma(g_{\beta,l}(\mathbf{c}_l)),$$

where $\mathbf{c}_l \in \mathbb{R}^d$ is the normalized residual context defined in Section 3, $g_{\beta,l}$ is a lightweight linear or two-layer branch, and σ is the logistic sigmoid. Thus $\beta_l \in (0, 2)$ in normal operation; the endpoints are saturated-logit limits.

2.2 Expanded Residual State

The matrix form separates persistent residual storage from sublayer compute. A standard Transformer uses the same width d both to store the residual stream and to feed attention and MLP blocks. DDL may instead maintain $\mathbf{X}_l \in \mathbb{R}^{B\times T\times d\times d_v}$. A learned compressor maps this state to $\mathbf{x}_l^{\text{in}} \in \mathbb{R}^d$ for each token; a standard width-d sublayer processes that vector; and the resulting direction, target, and gate update the persistent expanded state through Eq. (2.2). The protocol is therefore *Compress–Process–Rewrite*.

This design does not widen attention keys, queries, values, or MLP hidden activations from d to dd_v. It does increase residual-state capacity, activation traffic, and the cost of compression and rewriting. Expanded-state DDL is consequently a joint architectural change, not an isolation of the delta rewrite: without a matched expanded-state additive control, gains in this setting cannot be assigned uniquely to the read-and-erase term.

2.3 Spectral Analysis

We isolate the direct shortcut in Eq. (2.1) to describe its local geometry. The complete layer is state-dependent because $\mathbf{k}_l$, $\mathbf{v}_l$, and β_l are generated from the input. The following spectrum therefore applies only after conditioning on one layer, token, and residual state; it is not the Jacobian spectrum of the full nonlinear block.

Proposition 2.1 (Frozen shortcut spectrum)**.** Let $\mathbf{A} = \mathbf{I} - \beta\mathbf{k}\mathbf{k}^\top$, where $\mathbf{k} \in \mathbb{R}^d$ is a unit vector and $\beta \in \mathbb{R}$ is fixed. If $\beta \neq 0$, the eigenvalues of $\mathbf{A}$ are 1 with multiplicity $d - 1$ and $1 - \beta$ with multiplicity 1. The eigenvector for $1 - \beta$ is $\mathbf{k}$, and the eigenspace for eigenvalue 1 is $\mathbf{k}^\perp = \{\mathbf{u} \in \mathbb{R}^d : \mathbf{k}^\top\mathbf{u} = 0\}$. If $\beta = 0$, then $\mathbf{A} = \mathbf{I}$ and the eigenspace for eigenvalue 1 is all of $\mathbb{R}^d$.

Proof. For any $\mathbf{u} \in \mathbf{k}^\perp$, $\mathbf{A}\mathbf{u} = \mathbf{u} - \beta\mathbf{k}(\mathbf{k}^\top\mathbf{u}) = \mathbf{u}$. Also, $\mathbf{A}\mathbf{k} = \mathbf{k} - \beta\mathbf{k}(\mathbf{k}^\top\mathbf{k}) = (1 - \beta)\mathbf{k}$. When $\beta \neq 0$, these d independent eigen-directions span $\mathbb{R}^d$; when $\beta = 0$, the claim reduces to $\mathbf{A} = \mathbf{I}$. □

For any $\mathbf{u} = \mathbf{u}_\perp + (\mathbf{k}^\top\mathbf{u})\mathbf{k}$ with $\mathbf{u}_\perp \in \mathbf{k}^\perp$,

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

Thus the frozen shortcut leaves $\mathbf{k}^{\perp}$ unchanged and scales only the selected direction. For a matrix-valued residual state, the same left-multiplying operator acts on each of the d_v value columns; under vectorization, the shortcut is $\mathbf{I}_{d_v} \otimes \mathbf{A}$.

Together with the synchronized write, the selected readout evolves as

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

which yields three local regimes:

- $\beta \approx 0$**: skip.** The shortcut approaches $\mathbf{I}$, the write term vanishes, and the complete update approaches the identity.
- $\beta \approx 1$**: target match.** The old readout along $\mathbf{k}$ is removed and replaced by $\mathbf{v}^{\top}$, exactly so at $\beta = 1$.
- $\beta > 1$**: over-relaxed correction.** The readout crosses the target because $1 - \beta < 0$. At the saturated endpoint $\beta \to 2$, the direct shortcut—not the complete affine update—approaches the Householder reflector $\mathbf{I} - 2\mathbf{k}\mathbf{k}^{\top}$.

This analysis provides operator-level semantics for a conditioned local update: which one-dimensional subspace is edited, what target is requested, and how strongly the discrepancy is corrected. It does not imply that the learned direction $\mathbf{k}_l$ corresponds to a human-readable semantic feature.

3 DDL Transformer

We evaluate DDL in decoder-only Transformer language models with pre-norm RMSNorm, RoPE multi-head attention, and SwiGLU MLPs. DDL changes the residual interface while preserving the attention and MLP compute width d. For both scalar and expanded states, let $\mathbf{x}_l^{\mathrm{in}} \in \mathbb{R}^d$ denote the vector presented to a standard sublayer and define

$$\begin{aligned}\mathbf{c}_l &= \mathrm{RMSNorm}(\mathbf{x}_l^{\mathrm{in}}),\\ \mathbf{h}_l &= \mathbf{F}_l(\mathbf{c}_l).\end{aligned}$$

The reported configurations use the standard sublayer output as the unnormalized rewrite direction,

$$\widetilde{\mathbf{k}}_l = \mathbf{h}_l, \qquad \mathbf{k}_l = \frac{\widetilde{\mathbf{k}}_l}{\|\widetilde{\mathbf{k}}_l\|_2},$$

and generate the target and gate from the normalized residual context,

$$\mathbf{v}_l = W_{v,l}\mathbf{c}_l, \qquad \beta_l = 2\sigma(g_{\beta,l}(\mathbf{c}_l))\,.$$

Thus the reported models do not introduce a separate high-capacity subnetwork for $\mathbf{k}_l$: the ordinary attention or MLP sublayer chooses the edit direction, while lightweight branches produce $\mathbf{v}_l$ and β_l. Unless explicitly marked "w/o EC", expanded-state variants use embedding convolution (EC). Bare DDL denotes DDL-CC in the experimental sections.

3.1 Scalar residual state: $d_v = 1$

For $d_v = 1$, $\mathbf{X}_l$ reduces to $\mathbf{x}_l \in \mathbb{R}^d$, $\mathbf{x}_l^{\text{in}} = \mathbf{x}_l$, and Eq. (2.2) becomes

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

This setting tests the structured residual update without adding residual value channels or a compressor. It still adds direction normalization and the lightweight value and gate branches, so it is not compute-identical to the baseline.

Precision-friendly normalization. For low-precision training, we implement the unit-direction constraint through RMS normalization and a fixed scale $k_{\text{scale}} = 1/\sqrt{d}$:

$$\widehat{\mathbf{k}}_l = \mathrm{RMSNorm}(\widetilde{\mathbf{k}}_l; \epsilon_k^2/d), \qquad \mathbf{k}_l = \widehat{\mathbf{k}}_l/\sqrt{d}.$$

With $\epsilon_k = 0$, this is exact L_2 normalization. With $\epsilon_k > 0$, it is equivalent to

$$\mathbf{k}_l = \frac{\widetilde{\mathbf{k}}_l}{\sqrt{\|\widetilde{\mathbf{k}}_l\|_2^2 + \epsilon_k^2}},$$

so the unit-vector analysis is accurate whenever $\|\widetilde{\mathbf{k}}_l\|_2 \gg \epsilon_k$.

3.2 Expanded residual state: $d_v > 1$

For expanded-state DDL, $\mathbf{X}_l \in \mathbb{R}^{B\times T\times d\times d_v}$; we evaluate $d_v = 4$. EC initializes this state with a learnable depthwise causal short convolution that maps token embeddings into the d_v value channels. In the no-EC ablations, the token embedding is repeated across value channels:

$$\mathbf{X}_0 = \mathbf{x}_{\text{emb}}\mathbf{1}_{d_v}^\top.$$

The layer follows a Compress–Process–Rewrite protocol:

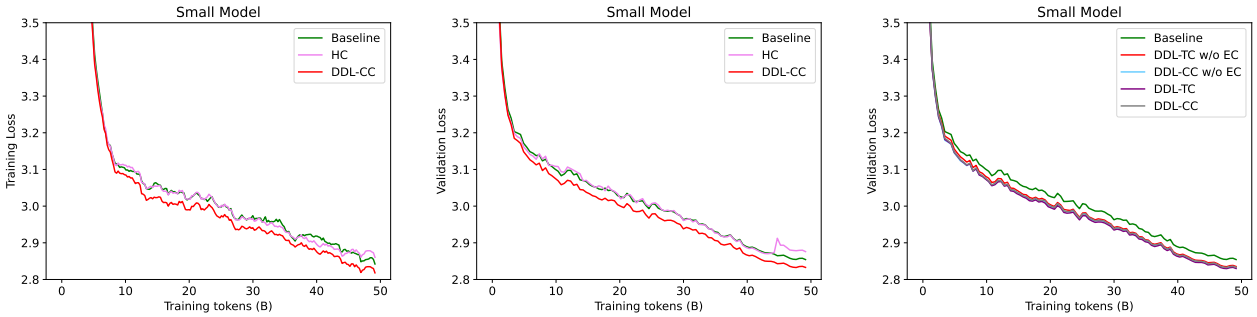
1. **Compress.** Map $\mathbf{X}_l$ to $\mathbf{x}_l^{\text{in}} \in \mathbb{R}^d$. DDL-TC applies a short causal depthwise convolution over the token dimension independently for each expanded channel and then pools across d_v. DDL-CC applies ordinary learned channel-axis mixing across the d_v values at the current token. We use CC as an implementation choice; channel mixing itself is not a methodological contribution.
2. **Process.** Apply the standard width-d attention or MLP sublayer to $\mathbf{c}_l = \mathrm{RMSNorm}(\mathbf{x}_l^{\text{in}})$, producing $\mathbf{h}_l = \mathbf{F}_l(\mathbf{c}_l)$.
3. **Rewrite.** Set $\widetilde{\mathbf{k}}_l = \mathbf{h}_l$, normalize it to $\mathbf{k}_l$, generate $\mathbf{v}_l = W_{v,l}\mathbf{c}_l$ and $\beta_l = 2\sigma(g_{\beta,l}(\mathbf{c}_l))$, and update $\mathbf{X}_l$ using Eq. (2.2).

The read-compare-write operation adds $O(dd_v)$ work and activation traffic per token and layer. The compressor adds $O(kdd_v)$ work for DDL-TC with token-axis kernel size k, and $O(dd_v)$ work for DDL-CC when its channel-axis kernel consumes all d_v values. The additional persistent state can be bandwidth- and memory-relevant even though attention and MLP widths remain d. During autoregressive generation, DDL-TC also needs a short per-layer history of expanded residual states; DDL-CC mixes only the current token's value channels and therefore avoids this additional token-history cache.

We report DDL-TC and DDL-CC with EC enabled by default, together with DDL-TC w/o EC and DDL-CC w/o EC. DDL-CC is denoted by bare DDL because it gives the strongest overall quality–efficiency compromise among the measured expanded-state implementations. These variants compare implementation choices within expanded-state DDL; they do not provide a matched expanded-state additive control for isolating the erase term.

4 Experiments

We compare a nanoGPT-based additive residual baseline (Karpathy, 2022) with scalar DDL ($d_v = 1$) and expanded-state DDL ($d_v = 4$). The expanded implementations differ in the compression axis: DDL-TC uses depthwise convolution (Chollet, 2017) over sequence positions, whereas DDL-CC uses learned mixing over the value-channel axis. EC denotes the default embedding-convolution input expansion; rows marked w/o EC instead repeat the token embedding across value channels. Bare DDL refers to DDL-CC. All reported model comparisons use the same training-token budget and one run per configuration; they are not iso-FLOPs comparisons.



(a) Small (124M) Train Loss **(b)** Small (124M) Val Loss **(c)** Small DDL Variants Val Loss

Figure 2 Small-scale loss curves for the baseline, DDL, and DDL variants trained on FineWeb-Edu for 49.15B tokens.

Table 1 1-shot evaluation results for small models trained on FineWeb-Edu for 49.15B tokens and evaluated with lm-evaluation-harness. The best and second-best scores in each column are bolded and underlined, respectively. Abbreviations: WG = WinoGrande. For expanded-state DDL variants, the default value of d_v is 4, and EC is enabled unless the row is marked w/o EC. Bare DDL refers to DDL-CC.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	<u>29.01</u>	55.85	37.59	30.20	<u>65.94</u>	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	<u>52.01</u>	48.73
DDL-TC w/o EC	27.90	<u>58.16</u>	38.26	30.80	**66.49**	80.30	**38.54**	50.83	48.91
DDL-CC w/o EC	27.82	**58.92**	**38.44**	<u>33.20</u>	65.83	79.50	38.28	51.07	49.13
DDL-TC	28.75	57.37	<u>38.41</u>	**34.40**	64.47	<u>82.00</u>	38.38	<u>52.01</u>	**49.47**
DDL-CC	28.33	57.87	38.24	32.20	64.09	**82.60**	<u>38.43</u>	**52.57**	<u>49.29</u>

4.1 Experimental settings

We train on FineWeb-Edu (Lozhkov et al., 2024). Each run uses 100,000 optimization steps, a global batch of 480 sequences, and sequence length 1,024, giving 491,520 tokens per update and

49.15B training tokens in total. We evaluate approximately 124M-parameter and 353M-parameter Llama-style models with RoPE (Su et al., 2024), SwiGLU activations (Shazeer, 2020), and query/key normalization.

All methods use the same μP-style parameterization and training recipe (Yang et al., 2022); we do not perform exhaustive per-method hyperparameter tuning. The learning rate is $1\mathrm{e}{-3}$ with cosine decay 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. Standard backbone bias terms are disabled and dropout is 0.0; the DDL gate retains the explicitly initialized output bias described in Appendix A. Each run uses four NVIDIA H200 GPUs. Other hyperparameters are listed in Appendix C. Because each configuration is represented by one training run, the tables report point estimates rather than means over seeds.

4.2 Experimental results

Figures 2, 4, and 3 show the available training and validation curves, and Table 3 reports final validation loss and perplexity. Scalar DDL gives lower final validation loss than the baseline in both runs: 2.8482 versus 2.8543 at the small scale and 2.6039 versus 2.6053 at the medium scale, corresponding to reductions of 0.0061 and 0.0014. These point differences are directionally consistent but small, especially at the medium scale, and cannot be distinguished from training-seed variation with the present single-run design.

Expanded-state variants give larger reductions. The best small-scale loss is 2.8299, a reduction of 0.0244 from the baseline, and the best medium-scale loss is 2.5758, a reduction of 0.0295. These models jointly introduce $d_v = 4$ residual storage, a compressor, and—unless marked w/o EC—embedding convolution. The no-EC rows show that EC is not necessary for a positive point improvement, but they do not isolate the delta rewrite from expanded residual capacity or compression.

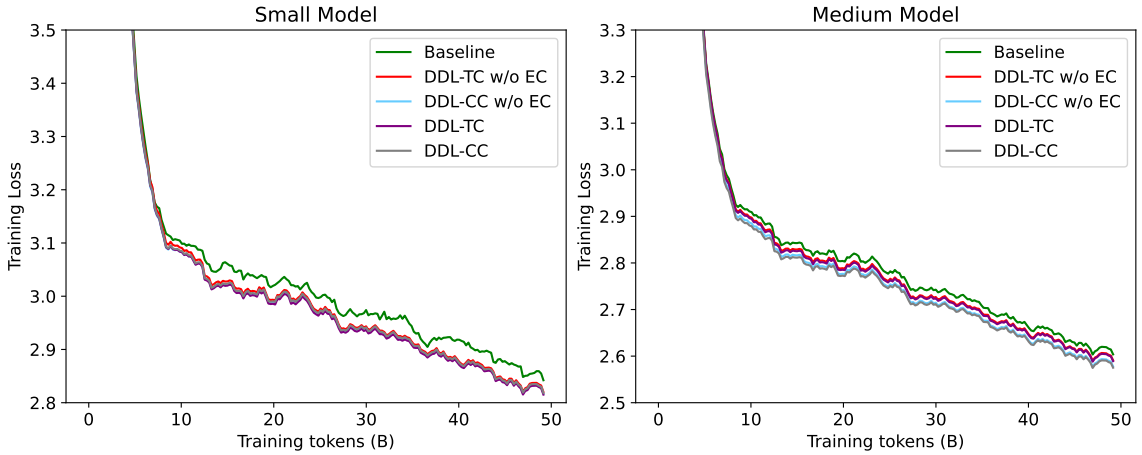
We evaluate one-shot and zero-shot performance on ARC (Yadav et al., 2019), HellaSwag (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), Social IQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021) using `lm-evaluation-harness` (Gao et al., 2021). Tables 1 and 2 report one-shot results; Appendix D reports zero-shot results. Scalar DDL raises the one-shot average by 0.17 and 0.73 points at the two scales. The best expanded-state averages improve by 0.91 and 1.18 points. Zero-shot averages are mixed across implementations, so we treat downstream evaluations as secondary evidence rather than a claim of uniform task-level dominance.

Tables 4 and 5 report hardware-specific throughput and peak memory. Expanded-state models trade lower validation loss for lower throughput and higher memory use. These measurements are useful operationally but do not replace total-FLOP or iso-FLOPs comparisons.

4.3 Expanded-state implementation variants

We report two $d_v = 4$ compressors.[1] DDL-TC mixes locally over sequence positions before pooling across value channels; DDL-CC performs learned channel mixing at the current token. Both use EC by default, and the w/o EC rows ablate only the input expansion. We select DDL-CC as the default implementation because it offers the strongest overall quality–efficiency compromise among the measured variants, not because channel-axis mixing is itself novel. Figures 2, 4, and 3 show the

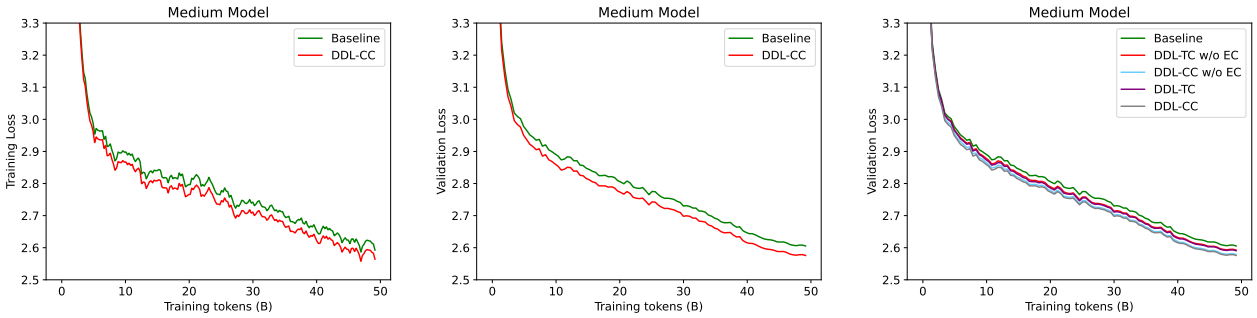
[1] Detailed implementations are provided in Appendix A.2.



(a) Small DDL Variants Train Loss **(b)** Medium DDL Variants Train Loss

Figure 3 Training loss curves for DDL implementation variants at small (∼124M) and medium (∼353M) scales, trained on FineWeb-Edu for 49.15B tokens.

corresponding loss curves; Tables 1 and 2 report one-shot results, and Appendix Tables 7 and 8 report zero-shot results.



(a) Medium (353M) Train Loss **(b)** Medium (353M) Val Loss **(c)** Medium DDL Variants Val

Figure 4 Medium-scale loss curves for the baseline, DDL, and DDL variants trained on FineWeb-Edu for 49.15B tokens.

Table 2 1-shot evaluation results for medium models trained on FineWeb-Edu for 49.15B tokens and evaluated with lm-evaluation-harness. The best and second-best scores in each column are bolded and underlined, respectively. Abbreviations: WG = WinoGrande. For expanded-state DDL variants, the default value of d_v is 4, and EC is enabled unless the row is marked w/o EC. Bare DDL refers to DDL-CC.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	33.62	**67.05**	47.42	33.20	<u>70.24</u>	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-TC w/o EC	33.02	<u>66.16</u>	47.83	35.60	69.86	<u>89.50</u>	**40.99**	55.64	54.83
DDL-CC w/o EC	34.30	66.08	<u>48.65</u>	**36.60**	68.72	88.80	<u>40.84</u>	55.33	<u>54.92</u>
DDL-TC	<u>35.15</u>	65.70	47.74	35.40	69.04	88.60	40.63	**56.59**	54.86
DDL-CC	34.39	65.57	**48.92**	<u>36.00</u>	69.48	**90.50**	40.53	<u>55.72</u>	**55.14**

Table 3 Validation loss and perplexity for small and medium models at the final step after 49.15B training tokens. The best loss and perplexity in each column are bolded.

Model	Small		Medium	
	Valid Loss	Valid Perplexity	Valid Loss	Valid Perplexity
Baseline	2.8543	17.3616	2.6053	13.5356
DDL ($d_v = 1$)	2.8482	17.2562	2.6039	13.5161
DDL-TC w/o EC	2.8355	17.0381	2.5927	13.3654
DDL-CC w/o EC	2.8321	16.9811	2.5790	13.1834
DDL-TC	**2.8299**	**16.9438**	2.5905	13.3370
DDL-CC	2.8329	16.9947	**2.5758**	**13.1420**

Table 4 Small-model point estimates for quality, throughput, and peak memory. Throughput and memory are measured on the same hardware and software stack; the peak-memory factor is normalized to the baseline. Optimized Triton kernels are used for CC and EC-enabled implementations. Expanded-state rows use $d_v = 4$ and EC unless marked w/o EC; bare DDL denotes DDL-CC. Total training FLOPs are not reported, so this table is not an iso-FLOPs comparison.

Model	Params	Val loss	Avg eval	Train tok/s (K)	Inference tok/s (K)	Peak memory	Peak-memory factor
Baseline	123M	2.8543	48.56	1509.6	1826.1	2.94GB	1.00×
DDL ($d_v = 1$)	123M	2.8482	48.73	1330.8	1605.8	2.94GB	1.00×
DDL-TC w/o EC	123M	2.8355	48.91	673.2	688.8	3.84GB	1.31×
DDL-CC w/o EC	123M	2.8321	49.13	1019.8	1043.0	3.47GB	1.18×
DDL-TC	123M	2.8299	49.47	783.5	865.1	3.38GB	1.15×
DDL-CC	123M	2.8329	49.29	1158.0	1220.7	3.08GB	1.05×

Table 5 Medium-model point estimates for quality, throughput, and peak memory. Throughput and memory are measured on the same hardware and software stack; the peak-memory factor is normalized to the baseline. Optimized Triton kernels are used for CC and EC-enabled implementations. Expanded-state rows use $d_v = 4$ and EC unless marked w/o EC; bare DDL denotes DDL-CC. Total training FLOPs are not reported, so this table is not an iso-FLOPs comparison.

Model	Params	Val loss	Avg eval	Train tok/s (K)	Inference tok/s (K)	Peak memory	Peak-memory factor
Baseline	353M	2.6053	53.96	537.1	531.5	7.06GB	1.00×
DDL-TC w/o EC	354M	2.5927	54.83	239.7	234.6	7.68GB	1.09×
DDL-CC w/o EC	353M	2.5790	54.92	358.5	337.7	7.45GB	1.06×
DDL-TC	354M	2.5905	54.86	282.9	291.1	7.40GB	1.05×
DDL-CC	353M	2.5758	55.14	422.3	400.5	7.20GB	1.02×

5 Scope of the Empirical Evidence

Training variance. Every configuration is represented by one training run. Scalar DDL is numerically better than the baseline in final validation loss and one-shot average at both scales, but the margins are small and no per-seed variance is available. We therefore do not claim that the scalar improvements are statistically resolved.

Attribution in the expanded state. The $d_v = 4$ models change residual capacity, compression, and sometimes input expansion together with the update rule. The no-EC ablations isolate EC, but not the value of the read-and-erase term. A matched control should preserve d_v, EC, the compressor, the $\mathbf{k}_l/\mathbf{v}_l/\beta_l$ branches, initialization, data order, and optimizer, while replacing Eq. (2.2) with the

write-only update

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

Without this control, the expanded-state results support the combined architecture but do not identify the causal contribution of delta rewriting alone.

Compute matching. All models are trained for the same number of tokens. We report measured throughput and peak memory for the available configurations, but not forward/backward FLOPs per token, total pretraining FLOPs, or end-to-end GPU-hours; the medium-scale cost table also lacks a scalar DDL measurement. Consequently, the current results do not establish an iso-FLOPs or iso-wall-clock advantage. Loss curves against cumulative FLOPs and elapsed time are needed to answer that question.

Operator-level rather than semantic interpretability. The update exposes a direction, target, discrepancy, and gate with exact local algebra, which makes the conditioned operator mechanically transparent. The present experiments do not audit the learned gate regimes or discrepancy reduction, and they do not show that $\mathbf{k}_l$ corresponds to a human-readable semantic concept. We therefore restrict the interpretability claim to the local operator, not the model's internal representations.

6 Related Work

DDL is most closely related to residual and gated pathways, delta-rule memory updates, expanded residual states, and low-rank shortcut transformations.

Residual and gated pathways. Highway Networks (Srivastava et al., 2015) introduced data-dependent gates around residual pathways, and later work explored richer cross-layer or dense residual connections (Chai et al., 2020; Menghani et al., 2025; Pagliardini et al., 2024; Fang et al., 2023; Xiao et al., 2025). DDL does not claim that a conventional residual block lacks the expressivity to implement replacement. Its distinction is the explicit form of the correction: a normalized rank-1 direction, a target readout, and a shared erase/write gate with an identity limit.

Delta rules and memory updates. The delta rule itself is established in efficient sequence models (Schlag et al., 2021; Yang et al., 2024). Those methods update a memory matrix over sequence time; DDL applies the same algebraic erase/write pattern over network depth and treats it as the residual interface between Transformer sublayers. Outer-product memory mechanisms (Mak and Flanigan, 2025) are related through low-rank writes. The contribution here is therefore the depth-wise architectural use and analysis of the rule, not invention of the delta rule.

Expanded residual states and compression. Hyper-Connections (Zhu et al., 2025) and manifold Hyper-Connections (Xie et al., 2025) expand or project residual streams to improve information flow. DDL's $d_v > 1$ construction is related in separating persistent state capacity from the width of the expensive sublayers. DDL-CC uses standard learned channel mixing as its compressor; we treat this as an implementation choice rather than a novel operation.

Orthogonal and low-rank transformations. Householder reflections are classical orthogonal transformations and have been used in neural architectures and adaptation methods (Yang et al., 2025; Dong et al., 2024; Arcas et al., 2025). Other work constrains weights or residual maps to be orthogonal or unitary for stability (Arjovsky et al., 2016; Jing et al., 2017; Zhang et al., 2021;

Fei et al., 2022; Wang et al., 2025; He et al., 2025). DDL does not impose global orthogonality. Only the frozen direct shortcut approaches a Householder reflector as $\beta \to 2$; the complete layer is input-dependent and affine after conditioning.

7 Conclusion

Deep Delta Learning parameterizes a Transformer residual correction as a target-seeking rank-1 read-compare-write update. Although a sufficiently expressive additive residual branch can represent the same map, DDL makes the selected readout, target, discrepancy, and step size explicit. This yields an identity limit, exact local target matching at $\beta = 1$, and a simple conditioned shortcut spectrum.

At approximately 124M and 353M parameters, all reported DDL configurations give lower final validation-loss point estimates than the baseline under an equal-token training budget. Scalar gains are small; expanded-state gains are larger but jointly reflect additional residual capacity, compression, and the DDL update. The measured results therefore motivate further study rather than establish complete mechanism attribution. The decisive next evidence is a matched expanded-state write-only control, multi-seed evaluation, compute-matched training curves, and an audit of how trained models use the gate and discrepancy. Within the scope of the present experiments, DDL provides a precise residual interface and a promising quality–efficiency tradeoff whose causal and statistical robustness remains to be resolved.

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.

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.

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, Yuxi Cai, Jintai Chen, Jingyu Zhao, Guangjian Tian, and Guodong Li. Cross-layer retrospective retrieving via layer attention. *arXiv preprint arXiv:2302.03985*, 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.

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.

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.

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

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.

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.

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 *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, 2021.

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 Implementation and Parameterization Details

Section 2 defines abstract generator functions $\mathcal{K}_l$, $\mathcal{V}_l$, and $\mathcal{B}_l$. This appendix states the concrete parameterization used by the reported Transformer models. The suffixes TC and CC denote compression choices for $d_v > 1$; EC is enabled by default and disabled only in rows marked w/o EC. Bare DDL denotes DDL-CC.

A.1 Reported Direction, Value, and Gate Generators

For each token and sublayer, let

$$\begin{aligned}\mathbf{x}_l^{\text{in}} &= \begin{cases}\mathbf{x}_l, & d_v = 1,\\ \mathcal{C}_l(\mathbf{X}_l), & d_v > 1,\end{cases}\\ \mathbf{c}_l &= \mathrm{RMSNorm}(\mathbf{x}_l^{\text{in}}),\\ \mathbf{h}_l &= \mathbf{F}_l(\mathbf{c}_l).\end{aligned}$$

Here $\mathcal{C}_l$ is the TC or CC compressor described below, and $\mathbf{F}_l$ is the standard attention or MLP sublayer.

Direction. The reported configurations set

$$\widetilde{\mathbf{k}}_l = \mathbf{h}_l, \qquad \mathbf{k}_l = \frac{\widetilde{\mathbf{k}}_l}{\sqrt{\|\widetilde{\mathbf{k}}_l\|_2^2 + \epsilon_k^2}}.$$

Thus there is no independent MLP that predicts $\mathbf{k}_l$ from pooled residual statistics in the reported experiments. The standard sublayer output determines the edit direction, and the small ϵ_k guard implements the precision-friendly normalization described in Section 3.1.

Value. The target vector is produced from the same normalized context presented to the backbone:

$$\mathbf{v}_l = W_{v,l}\mathbf{c}_l \in \mathbb{R}^{d_v}.$$

For $d_v = 1$, this is a scalar target. For $d_v = 4$, it specifies the desired readout in each residual value channel. No additional attention or MLP block is used for the value branch.

Gate. The scalar gate is computed per token as

$$\beta_l = 2\sigma(W_{\beta,l}\mathbf{c}_l + b_{\beta,l})$$

or, when the configured two-layer branch is enabled,

$$\beta_l = 2\sigma(W_{\beta,2,l}\tanh(W_{\beta,1,l}\mathbf{c}_l + b_{\beta,1,l}) + b_{\beta,2,l})\,.$$

The implementation computes gate logits in fp32 for stability. The output bias is initialized to match the configured starting value $\beta_0 = \texttt{ddl\_beta\_init} \in [0, 2]$ through $\mathrm{logit}(\beta_0/2)$, with clamping in code. The same gate configuration is used across reported variants unless explicitly stated otherwise.

A.2 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-TC-EC.py`: user-facing DDL-TC. This is the default token-compression implementation of expanded-state DDL on top of GPT (MHA + RoPE). It expands token embeddings with a learnable depthwise causal short convolution (`input_embed_shortconv_kernel_size`, identity-initialized) and compresses the expanded residual with a token-axis short causal convolution plus a learned read vector.
- `model/DDL-gpt-mha-rope-TC.py`: user-facing DDL-TC w/o EC. This ablation keeps the token-axis residual compressor but initializes the state by repeating token embeddings across the value-channel axis.
- `model/DDL-gpt-mha-rope-CC-EC.py`: user-facing DDL-CC and the default implementation denoted by bare DDL in the main text. It expands token embeddings with a learnable depthwise causal short convolution and compresses the expanded residual along the value-channel axis d_v; the Conv consumes the value axis and returns length 1 (`ddl_state_shortconv_kernel_size` $= d_v$).
- `model/DDL-gpt-mha-rope-CC.py`: user-facing DDL-CC w/o EC. This ablation keeps the channel-axis residual compressor but initializes the state by repeating token embeddings across the value-channel axis.

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 default DDL-TC and DDL-CC modules, EC uses a depthwise causal short convolution over the token dimension to map $(B, T, d) \to (B, T, d \cdot d_v)$ and then reshapes to (B, T, d, d_v); the convolution is identity-initialized, so the starting behavior matches simple repetition. In the w/o EC ablations, 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)`).

ShortConv compression along T (DDL-TC and DDL-TC w/o EC). In `model/DDL-gpt-mha-rope-TC-EC.py` and `model/DDL-gpt-mha-rope-TC.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). During autoregressive generation, this token-axis convolution requires retaining the last $k-1$ expanded residual states per layer, in addition to the usual attention KV cache.

ShortConv compression along d_v (DDL-CC and DDL-CC w/o EC). In `model/DDL-gpt-mha-rope-CC-EC.py` and `model/DDL-gpt-mha-rope-CC.py`, we instead apply a depthwise 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. The EC-enabled CC implementation is the default DDL configuration in our runs because it provides the best overall quality–cost tradeoff. 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.

B Relation to DeltaNets and Householder Products

The delta rule is prior work rather than a new algebraic contribution of DDL. Our connection is to the DeltaNet architecture (Schlag et al., 2021), which replaces additive accumulation in Linear Transformers with a delta-rule memory update.

We spell out the algebraic analogy between Deep Delta Learning and 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 $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$:

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

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. (2.1) acting over depth l: Eq. (B.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).

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

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

This reveals a direct algebraic 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 shortcut operator with a rank-1 perturbation, $\mathbf{I} - \beta\, \mathbf{k}\mathbf{k}^\top$: 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 depth-wise delta-rule step size. This ensures that erasure and injection are modulated synchronously, preserving the identity limit of the complete update.

Thus, DDL applies an established delta-rule erase/write operation to layer-wise feature evolution. The novelty claim is restricted to using this rule as a depth-wise Transformer residual interface and analyzing and instantiating that interface.

C Hyper-parameter Settings

Table 6 lists the architecture hyperparameters for the small and medium model sizes.

Table 6 Architecture hyperparameters for the small and medium model sizes.

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

Tables 7 and 8 report zero-shot evaluation results for small and medium models. These single-run point estimates are mixed across implementations and are included as supplementary task-level measurements rather than uniform evidence of downstream improvement.

Table 7 0-shot evaluation results for small models trained on FineWeb-Edu for 49.15B tokens and evaluated with lm-evaluation-harness. The best and second-best scores in each column are bolded and underlined, respectively. Abbreviations: WG = WinoGrande. For expanded-state DDL variants, the default value of d_v is 4, and EC is enabled unless the row is marked w/o EC. Bare DDL refers to DDL-CC.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	<u>28.33</u>	<u>52.44</u>	37.60	33.00	<u>65.94</u>	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-TC w/o EC	27.30	**52.95**	38.40	**33.80**	65.40	**73.70**	<u>38.64</u>	50.12	<u>47.54</u>
DDL-CC w/o EC	27.05	51.47	<u>38.72</u>	32.20	**66.16**	71.80	38.08	51.07	47.07
DDL-TC	**28.50**	51.89	**38.82**	<u>33.20</u>	65.29	73.00	**39.05**	<u>52.88</u>	**47.83**
DDL-CC	27.90	51.26	38.42	31.40	64.85	<u>73.60</u>	37.77	52.25	47.18

Table 8 0-shot evaluation results for medium models trained on FineWeb-Edu for 49.15B tokens and evaluated with lm-evaluation-harness. The best and second-best scores in each column are bolded and underlined, respectively. Abbreviations: WG = WinoGrande. For expanded-state DDL variants, the default value of d_v is 4, and EC is enabled unless the row is marked w/o EC. Bare DDL refers to DDL-CC.

Model	ARC-C	ARC-E	Hellaswag	OpenBookQA	PIQA	SciQ	Social IQA	WG	Avg.
Baseline	31.74	<u>59.85</u>	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-TC w/o EC	32.08	58.38	48.08	35.80	69.42	79.90	39.92	54.14	52.22
DDL-CC w/o EC	32.08	**61.74**	**49.12**	34.80	69.53	80.20	<u>40.43</u>	<u>55.17</u>	**52.88**
DDL-TC	**33.02**	59.68	48.00	**36.80**	68.77	<u>81.00</u>	39.82	**55.88**	<u>52.87</u>
DDL-CC	<u>32.59</u>	59.55	<u>49.01</u>	<u>36.00</u>	<u>69.70</u>	**82.30**	39.82	53.83	52.85