DEEP DELTA LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer residual streams are updated by addition. A sufficiently expressive residual block can represent content replacement, but the residual update itself has no operation that reads, compares, and replaces content. We introduce **Deep Delta Learning (DDL)**, which applies the delta rule over network depth. Each layer reads the residual state along a learned direction, compares the readout with a learned target, and writes a gated rank-1 correction back along the same direction. A closed gate gives the identity map, and a unit gate overwrites the selected readout with the target. DDL works with the usual vector state or with an expanded state that stores several value channels, while attention and MLP blocks keep the original model width. We pretrain decoder-only models with approximately GPT-2 small and medium sizes on the FineWeb-Edu dataset. At both scales, every DDL variant has lower validation loss and higher average one-shot accuracy than the additive baseline, and the best expanded variants raise that average by 0.91 and 1.18 points.

1 INTRODUCTION

Residual connections stabilize deep networks by combining an identity path with an incremental transformation (He et al., 2016). In Transformer language models, they also carry the token-level state that attention and MLP blocks read and write, $\mathbf{x}_{l+1} = \mathbf{x}_l + \mathbf{F}_l(\mathbf{x}_l)$, where $\mathbf{x}_l \in \mathbb{R}^{B\times T\times d}$ for batch size B, sequence length T, and compute width d. A sufficiently expressive branch can replace content by cancelling an old value and inserting a new one. The addition itself, however, neither reads the old value nor specifies the new one; both steps stay implicit inside the branch.

We introduce Deep Delta Learning (**DDL**), a residual update that makes both steps explicit:

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

Each layer reads the state along a learned unit direction $\mathbf{k}_l$, compares the readout with a target $\mathbf{v}_l^\top$, and writes a gated rank-1 correction along the same direction. One gate β_l scales both the erase and the write: $\beta_l = 0$ gives the identity, and $\beta_l = 1$ sets the selected readout exactly to the target. The update is still additive, so DDL changes how the correction is parameterized, not which functions the network can represent. Sequence-memory models apply the same delta rule over time (Schlag et al., 2021; Yang et al., 2024); DDL applies it over depth, between Transformer sublayers.

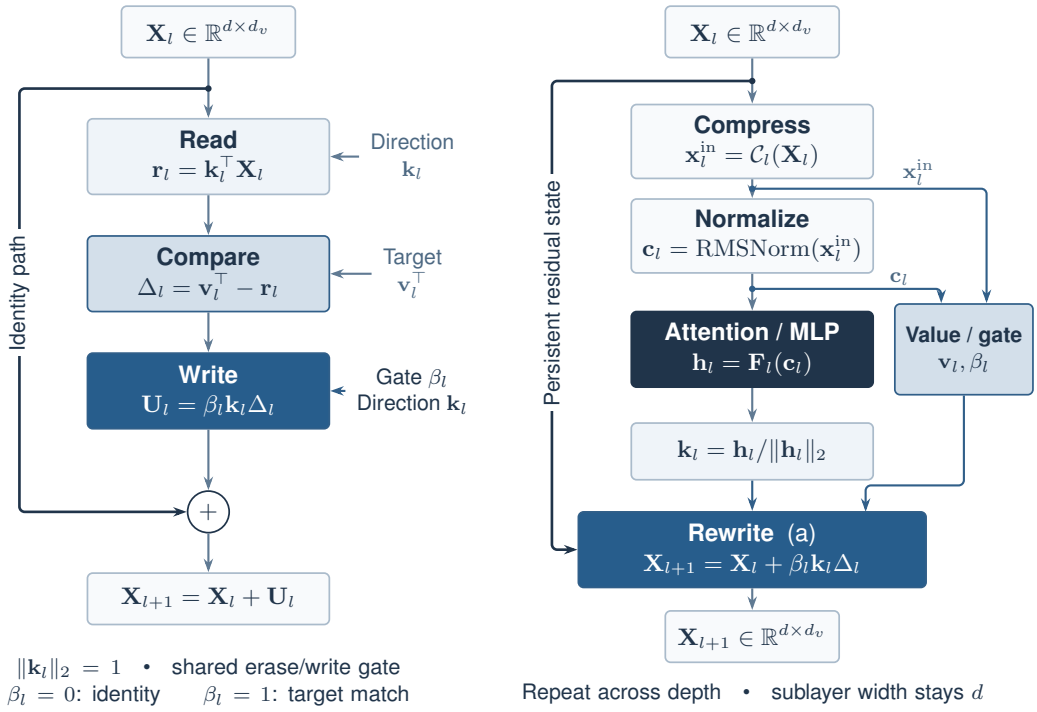
DDL works with the ordinary vector state ($d_v = 1$) or with an expanded state $\mathbf{X}_l \in \mathbb{R}^{B\times T\times d\times d_v}$. In the expanded case, a learned compressor reduces the state to a width-d input for each attention or MLP block. The residual state grows by a factor of d_v while those blocks stay at width d; the price is extra memory, bandwidth, and compression work.

We train decoder-only models of GPT-2 small and medium size (Radford et al., 2019) on 49.15B FineWeb-Edu tokens. At both sizes, scalar DDL reduces validation loss by 0.0061 and 0.0014 and raises average one-shot accuracy by 0.17 and 0.73 points. The best expanded variants reduce validation loss by 0.0244 and 0.0295 and raise the one-shot average by 0.91 and 1.18 points. Each configuration is trained once on the same token budget, so these comparisons neither establish statistical significance nor separate the erase term from the added residual capacity.

Our contributions are:

1. We propose **Deep Delta Learning (DDL)**, a delta-rule residual connection: the sublayer output sets the edit direction, and lightweight branches set the target and a gate β_l that scales the readout error by $1 - \beta_l$, from identity ($\beta_l = 0$) through exact overwrite ($\beta_l = 1$) to overshoot.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) The residual rewrite.

(b) A DDL Transformer sublayer.

Figure 1: **Deep Delta Learning overview.** (a) Read the selected residual content, compare it with a target, and add a gated rank-1 correction to the identity path. (b) A width-d sublayer produces the direction; lightweight branches produce the target from the sublayer input $\mathbf{x}_l^{\text{in}}$ and the gate from the normalized context $\mathbf{c}_l$. The expanded residual persists across sublayers. Batch and sequence axes are omitted.

2. We extend DDL to an expanded residual state with d_v value channels through a Compress–Process–Rewrite interface: a learned compressor reads the state into a width-d input, the attention or MLP sublayer runs at its original width, and one rank-1 delta update rewrites all d_v channels.
3. We pretrain scalar DDL and four expanded-state variants (two compressors, each with and without embedding convolution) against an additive baseline at GPT-2 small and medium sizes on FineWeb-Edu. Every variant lowers validation loss and raises the one-shot average over eight tasks, and at both sizes every expanded variant improves on scalar DDL in both metrics.

2 Deep Delta Learning

We derive the update for one token and 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 the number of value channels; $d_v = 1$ recovers a vector state.

2.1 Delta Residual Rewrite

Given $\mathbf{X}_l$, three generator functions produce an unnormalized direction $\tilde{\mathbf{k}}_l = \mathcal{K}_l(\mathbf{X}_l) \in \mathbb{R}^d$, a target value $\mathbf{v}_l = \mathcal{V}_l(\mathbf{X}_l) \in \mathbb{R}^{d_v}$, and a gate $\beta_l = \mathcal{B}_l(\mathbf{X}_l) \in (0, 2)$. These functions are instantiated in Section 3: $\mathcal{K}_l$ contains the standard sublayer, while $\mathcal{V}_l$ and $\mathcal{B}_l$ are lightweight branches on the sublayer input.

Read/write direction. We normalize the proposed direction to $\mathbf{k}_l = \tilde{\mathbf{k}}_l / \|\tilde{\mathbf{k}}_l\|_2$, so $\|\mathbf{k}_l\|_2 = 1$, 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 $\beta_l = 1$, $\mathbf{k}_l^\top \mathbf{X}_{l+1} = \mathbf{v}_l^\top$, 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 $\mathbf{e}_l^{\mathrm{pre}} = \mathbf{k}_l^\top \mathbf{X}_l - \mathbf{v}_l^\top$ and $\mathbf{e}_l^{\mathrm{post}} = \mathbf{k}_l^\top \mathbf{X}_{l+1} - \mathbf{v}_l^\top$. Using $\|\mathbf{k}_l\|_2 = 1$ in Eq. (2.2) gives $\mathbf{e}_l^{\mathrm{post}} = (1 - \beta_l)\mathbf{e}_l^{\mathrm{pre}}$. 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.

The sigmoid gate used below yields $\beta_l \in (0, 2)$; endpoints are saturated-logit limits.

2.2 Expanded Residual State

For $d_v > 1$, DDL separates persistent storage from sublayer compute through a *Compress–Process–Rewrite* interface: compress $\mathbf{X}_l$ to a width-d vector, process it with attention or an MLP, and rewrite the expanded state via Eq. (2.2). Attention keys, queries, values, and MLP hidden activations are not widened to dd_v.

2.3 Spectral Analysis

The spectrum below conditions on the generated direction and gate for a fixed token and state. It describes the direct shortcut, not the Jacobian of the full input-dependent layer.

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 $\mathbf{I}_{d_v} \otimes \mathbf{A}$. Since $\det \mathbf{A} = 1 - \beta$, the conditioned update $\mathbf{X} \mapsto \mathbf{A}\mathbf{X} + \beta \mathbf{k}\mathbf{v}^\top$ is an invertible affine map for every $\beta \neq 1$; at $\beta = 1$, $\mathbf{A}$ is the orthogonal projector onto $\mathbf{k}^\perp$, and the old readout along $\mathbf{k}$ is discarded.

Together with the synchronized write, the selected readout evolves as $\mathbf{k}^\top \mathbf{X}_{l+1} = \mathbf{v}^\top + (1 - \beta)(\mathbf{k}^\top \mathbf{X}_l - \mathbf{v}^\top)$, the error contraction above, 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 approaches the Householder reflector $\mathbf{I} - 2\mathbf{k}\mathbf{k}^\top$; the complete affine update does not.

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.

2.4 Relation to DeltaNet

DDL takes its update from the delta rule of DeltaNet (Schlag et al., 2021), which replaces additive accumulation in linear Transformers with a delta-rule memory update; the rule is prior work, not an algebraic contribution of DDL. Written with left multiplication, DeltaNet updates a memory $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ over time t as

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top, \tag{2.3}$$

which is the more common update $\mathbf{M}_t = \mathbf{M}_{t-1} + \beta_t(\mathbf{v}_t - \mathbf{M}_{t-1}\mathbf{k}_t)\mathbf{k}_t^\top$ with $\mathbf{S}_t = \mathbf{M}_t^\top$. Eq. (2.3) and the DDL update Eq. (2.1) correspond term by term. The memory $\mathbf{S}_t$ corresponds to the residual state $\mathbf{X}_l$, and the key dimension d_k to the feature dimension d. Both apply the shortcut $\mathbf{I} - \beta\mathbf{k}\mathbf{k}^\top$, which for $\|\mathbf{k}\|_2 = 1$ is an orthogonal projector at $\beta = 1$ and a Householder reflection at $\beta = 2$; DeltaNet applies it over time steps, DDL over network depth. Both write $\beta\mathbf{k}\mathbf{v}^\top$, so the same β scales erasure and writing and acts as a step size, which gives the complete DDL update its identity limit at $\beta_l = 0$.

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 $\mathbf{x}_l^{\text{in}} \in \mathbb{R}^d$ denote the vector presented to a standard sublayer, $\mathbf{c}_l = \mathrm{RMSNorm}(\mathbf{x}_l^{\text{in}})$ its normalized form, and $\mathbf{h}_l = \mathbf{F}_l(\mathbf{c}_l)$ the sublayer output. The reported configurations use $\mathbf{h}_l$ as the unnormalized rewrite direction, $\tilde{\mathbf{k}}_l = \mathbf{h}_l$ and $\mathbf{k}_l = \tilde{\mathbf{k}}_l / \|\tilde{\mathbf{k}}_l\|_2$, and generate the target from the sublayer input and the gate from its normalized form,

$$\mathbf{v}_l = 4\,\sigma\big(W_{v,l}\mathbf{x}_l^{\text{in}} + \mathbf{b}_{v,l}\big)\,, \qquad \beta_l = 2\sigma(W_{\beta,l}\mathbf{c}_l + b_{\beta,l})\,,$$

where σ is the elementwise logistic sigmoid, so each target entry lies in $(0, 4)$. Gate logits are computed in fp32, and $b_{\beta,l}$ is initialized to 0, so every gate starts at $\beta_l \approx 1$, the target-match regime. 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. These branches add $(d_v + 1)(d + 1)$ parameters per sublayer, and at the small scale every DDL variant stays within 0.3M parameters of the baseline (Table 7).

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 $\mathbf{x}_{l+1} = \mathbf{x}_l + \beta_l(v_l - \mathbf{k}_l^\top \mathbf{x}_l)\mathbf{k}_l$. 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 $k_{\text{scale}} = 1/\sqrt{d}$: $\hat{\mathbf{k}}_l = \mathrm{RMSNorm}(\tilde{\mathbf{k}}_l; \epsilon_k^2/d)$ and $\mathbf{k}_l = \hat{\mathbf{k}}_l/\sqrt{d}$. With $\epsilon_k = 0$, this is exact L_2 normalization. With $\epsilon_k > 0$, it is equivalent to $\mathbf{k}_l = \tilde{\mathbf{k}}_l / \sqrt{\|\tilde{\mathbf{k}}_l\|_2^2 + \epsilon_k^2}$, so the unit-vector analysis is accurate whenever $\|\tilde{\mathbf{k}}_l\|_2 \gg \epsilon_k$.

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 $d_v = 4$. In the no-EC ablations, every value channel starts as the token embedding, $\mathbf{X}_0 = \mathbf{x}_{\text{emb}} \mathbf{1}_{d_v}^\top$. Embedding convolution (EC) instead

maps each embedding feature to d_v channels with a learnable causal depthwise convolution over tokens (Chollet, 2017),

$$\mathbf{X}_{0,t,i,j} = \sum_{s=0}^{k_e-1} e_{i,j,s}\, \mathbf{x}_{\mathrm{emb},t-s,i},$$

with kernel size $k_e = 4$, initialized to $e_{i,j,0} = 1$ and $e_{i,j,s} = 0$ for $s > 0$ so that EC starts as repetition.

The layer follows a Compress–Process–Rewrite protocol:

1. **Compress.** Map $\mathbf{X}_l$ to $\mathbf{x}_l^{\mathrm{in}} \in \mathbb{R}^d$ with the block's compressor $\mathcal{C}_l$, which is applied before the attention sublayer and again, after the attention update, before the MLP sublayer. DDL-TC mixes over the last k tokens and then across value channels; DDL-CC mixes only the value channels of the current token:

$$\text{TC: } \mathbf{x}_{t,i}^{\mathrm{in}} = \sum_{j=1}^{d_v} r_j \sum_{s=0}^{k-1} c_{i,j,s}\, \mathbf{X}_{t-s,i,j}, \qquad \text{CC: } \mathbf{x}_{t,i}^{\mathrm{in}} = \sum_{j=1}^{d_v} w_{i,j}\, \mathbf{X}_{t,i,j}.$$

   Both use kernel size $k = 4$ and a read vector $\mathbf{r} \in \mathbb{R}^{d_v}$. In CC, $w_{i,j} = \sum_{a=1}^{d_v} r_a\, c_{i,j-a+\lfloor (k-1)/2 \rfloor}$ comes from a zero-padded length-k kernel $c_{i,\cdot}$ along the value axis, with $c_{i,m} = 0$ outside $0 \le m < k$. Kernels start at $1/k$ and $\mathbf{r}$ at $1/d_v$, so TC starts as an average over the last k tokens and all value channels. A separate compressor reads the final state out before the final RMSNorm and the language-model head.
2. **Process.** Apply the standard width-d attention or MLP sublayer to $\mathbf{c}_l = \mathrm{RMSNorm}(\mathbf{x}_l^{\mathrm{in}})$, producing $\mathbf{h}_l = \mathbf{F}_l(\mathbf{c}_l)$.
3. **Rewrite.** Set $\tilde{\mathbf{k}}_l = \mathbf{h}_l$, normalize it to $\mathbf{k}_l$, generate $\mathbf{v}_l$ and β_l as above, 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. The larger persistent state can still add memory and bandwidth cost, although attention and MLP widths remain d. During autoregressive generation, TC keeps the last $k-1$ expanded states at each compressor input and EC the last $k_e - 1$ token embeddings, in addition to the attention KV cache; CC needs no token history.

We report four expanded-state variants: DDL-TC and DDL-CC, which enable EC by default, and their ablations DDL-TC w/o EC and DDL-CC w/o EC. DDL-CC is the default expanded-state configuration because it gives the best overall quality–efficiency compromise among the measured implementations.

4 Experiments

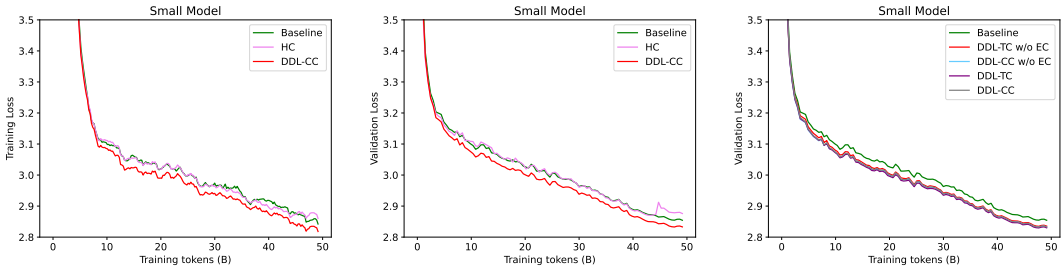
We compare a nanoGPT-based additive baseline (Karpathy, 2022) with scalar DDL ($d_v = 1$) and expanded DDL ($d_v = 4$). Each configuration is trained once with the same training-token budget, so the tables report point estimates rather than means over seeds; comparisons are not iso-FLOPs.

Table 1: One-shot accuracy (%) for small models. Bold/underlined entries are best/second-best; WG = WinoGrande. Expanded variants use $d_v = 4$ and EC unless marked w/o EC.

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 a sequence length 1,024, giving 491,520 tokens per update and



(a) Small Train Loss (b) Small Val Loss (c) Small DDL Variants Val Loss

Figure 2: Small-scale loss curves for the baseline and DDL variants. Panels (a) and (b) also show Hyper-Connections (HC) (Zhu et al., 2025) with expansion rate 4, which is not included in the tables.

49.15B training tokens in total. The models are Llama-style, with RoPE (Su et al., 2024), SwiGLU activations (Shazeer, 2020), and query/key normalization, and have the depth and width of GPT-2 small and medium (Table 2).

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 & 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 Section 3. Each run uses four NVIDIA H200 GPUs.

Table 2: Architecture hyperparameters for the small and medium model sizes.

Model	#Param	#Layer	#Head	Head Dimension	Hidden Size
Small-size Model	123.6M	12	6	128	768
Medium-size Model	353.5M	24	8	128	1024

4.2 Experimental results

Figures 2 and 3 show training and validation curves; Figure 4 gives training curves for the expanded-state variants, and Table 6 reports final validation loss and perplexity. Scalar DDL gives lower final validation loss than the baseline at both scales: 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 (DDL-TC), a reduction of 0.0244 from the baseline, and the best medium-scale loss is 2.5758 (DDL-CC), 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. Isolating it requires a matched control that keeps d_v, EC, the compressor, the $\mathbf{k}_l$, $\mathbf{v}_l$, and β_l branches, initialization, data order, and optimizer, and replaces Eq. (2.2) with the write-only update $\mathbf{X}_{l+1} = \mathbf{X}_l + \beta_l \mathbf{k}_l \mathbf{v}_l^\top$.

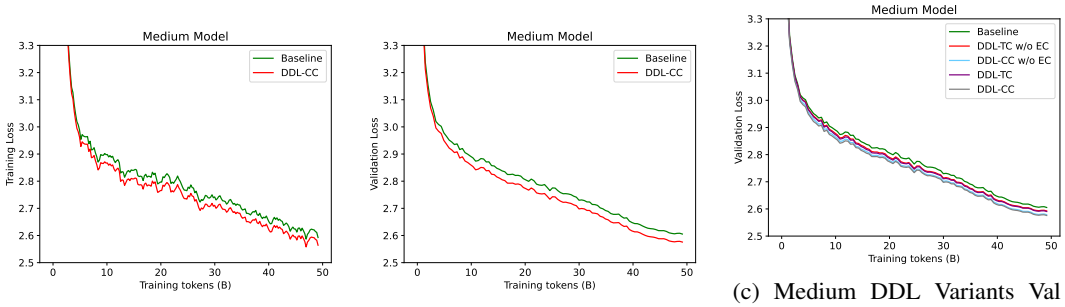
We evaluate one-shot and zero-shot performance on ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), Social IQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021) using `lm-evaluation-harness` (Gao et al., 2021). Tables 1 and 3 report one-shot results; Tables 4 and 5 report 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. No configuration is best on every task, and zero-shot averages are mixed across implementations: at the small scale, DDL-CC and DDL-CC w/o EC average below the baseline. We therefore treat downstream evaluations as secondary evidence.

Tables 7 and 8 report hardware-specific throughput and peak memory. Expanded-state models trade lower validation loss for lower throughput and higher memory use.

Quality–cost tradeoffs. At the small scale, scalar DDL retains the baseline's measured peak memory of 2.94 GB, but training throughput falls from 1509.6K to 1330.8K tokens/s and inference throughput from 1826.1K to 1605.8K tokens/s: without residual expansion, the model still pays for normalization and the additional branches. Among expanded models, DDL-TC achieves the lowest validation loss (2.8299) and highest one-shot average (49.47), whereas DDL-CC gives 2.8329 and 49.29 with substantially higher training throughput (1158.0K versus 783.5K tokens/s) and lower peak memory (3.08 versus 3.38 GB). We therefore use CC as the default compromise, although TC is better on both small-scale quality metrics.

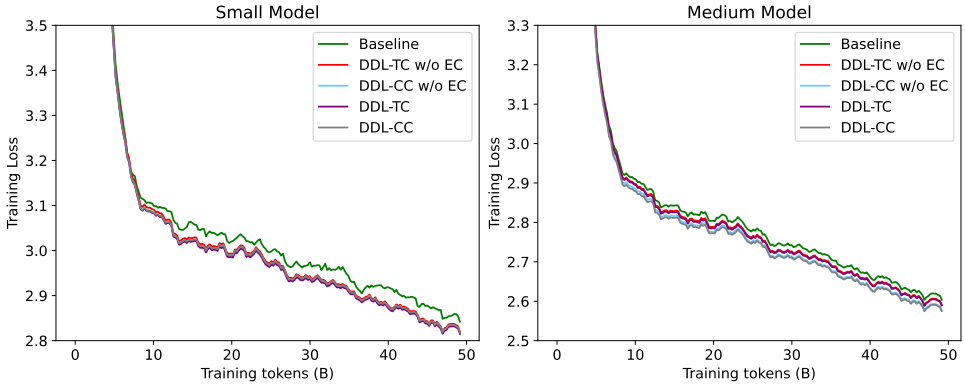
At the medium scale, DDL-CC has both the lowest validation loss (2.5758) and the highest one-shot average (55.14) among the reported configurations. Its training throughput is 422.3K tokens/s, compared with 537.1K for the baseline and 282.9K for DDL-TC; peak memory is 7.20 GB versus the baseline's 7.06 GB. EC also changes the measured execution cost: CC with EC is faster and uses less peak memory than CC w/o EC at both scales in the reported stack. Because CC and the EC-enabled implementations use optimized kernels, these differences reflect the measured implementations, not an implementation-independent complexity ordering. The medium-scale cost table lacks a scalar DDL measurement, so the cost comparison across state sizes is incomplete.

Effect of input expansion. EC's quality effect depends on the compressor and model scale. With TC, enabling EC lowers validation loss from 2.8355 to 2.8299 at the small scale and from 2.5927 to 2.5905 at the medium scale. With CC, small-scale validation loss instead rises slightly from 2.8321 to 2.8329, while the one-shot average increases from 49.13 to 49.29; at the medium scale, both metrics improve, from 2.5790 to 2.5758 and from 54.92 to 55.14.



(a) Medium Train Loss (b) Medium Val Loss (c) Medium DDL Variants Val Loss

Figure 3: Medium-scale loss curves for the baseline and DDL variants.



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

Figure 4: Training loss curves for the DDL variants at the small and medium scales.

Table 3: One-shot accuracy (%) for medium models. Bold/underlined entries are best/second-best; WG = WinoGrande. Expanded variants use $d_v = 4$ and EC unless marked w/o EC.

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 4: Zero-shot accuracy (%) for small models, evaluated with lm-evaluation-harness. Bold/underlined entries are best/second-best; WG = WinoGrande. Expanded variants use $d_v = 4$ and EC unless marked w/o EC.

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 5: Zero-shot accuracy (%) for medium models, evaluated with lm-evaluation-harness. Bold/underlined entries are best/second-best; WG = WinoGrande. Expanded variants use $d_v = 4$ and EC unless marked w/o EC.

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

Table 6: Final validation loss and perplexity for small and medium models. 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 7: Small-model quality and cost on the same hardware/software stack. Memory factors are relative to the baseline; CC and EC-enabled implementations use optimized Triton kernels. Expanded variants use $d_v = 4$. Total training FLOPs are not reported.

Model	Params	Val loss	Avg eval	Train tok/s (K)	Inference tok/s (K)	Peak memory	Peak-memory factor
Baseline	123.6M	2.8543	48.56	1509.6	1826.1	2.94GB	1.00×
DDL ($d_v = 1$)	123.6M	2.8482	48.73	1330.8	1605.8	2.94GB	1.00×
DDL-TC w/o EC	123.8M	2.8355	48.91	673.2	688.8	3.84GB	1.31×
DDL-CC w/o EC	123.7M	2.8321	49.13	1019.8	1043.0	3.47GB	1.18×
DDL-TC	123.9M	2.8299	49.47	783.5	865.1	3.38GB	1.15×
DDL-CC	123.7M	2.8329	49.29	1158.0	1220.7	3.08GB	1.05×

5 Related Work

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

Table 8: Medium-model quality and cost on the same hardware/software stack. Memory factors are relative to the baseline; CC and EC-enabled implementations use optimized Triton kernels. Expanded variants use $d_v = 4$. Total training FLOPs are not reported.

Model	Params	Val loss	Avg eval	Train tok/s (K)	Inference tok/s (K)	Peak memory	Peak-memory factor
Baseline	353.5M	2.6053	53.96	537.1	531.5	7.06GB	1.00×
DDL-TC w/o EC	354.2M	2.5927	54.83	239.7	234.6	7.68GB	1.09×
DDL-CC w/o EC	353.9M	2.5790	54.92	358.5	337.7	7.45GB	1.06×
DDL-TC	354.2M	2.5905	54.86	282.9	291.1	7.40GB	1.05×
DDL-CC	353.9M	2.5758	55.14	422.3	400.5	7.20GB	1.02×

residual connections (Chai et al., 2020; Menghani et al., 2025; Pagliardini et al., 2024; Fang et al., 2023; Xiao et al., 2025). DDL makes the correction explicit through 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 is established in efficient sequence models (Schlag et al., 2021; Yang et al., 2024), which update a memory matrix over sequence time; Section 2.4 relates it to DDL term by term. Our contribution is its depth-wise use as the residual interface between Transformer sublayers, and the analysis of that interface. Outer-product memory mechanisms (Mak & Flanigan, 2025) are related through low-rank writes.

Expanded residual states and compression. Hyper-Connections (Zhu et al., 2025) keep n copies of the residual stream and mix them with learned $n \times n$ matrices, and manifold-constrained Hyper-Connections (Xie et al., 2025) restrict that mixing matrix to be doubly stochastic. DDL's $d_v > 1$ construction shares their goal of separating persistent state capacity from the width of the expensive sublayers, but its shortcut acts on the feature axis: $\mathbf{I} - \beta_l \mathbf{k}_l \mathbf{k}_l^\top$ left-multiplies $\mathbf{X}_l$ and applies the same operator to every value channel. Its CC compressor is standard learned channel mixing.

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 $\beta \to 2$.

6 Conclusion

DDL makes each residual update a gated rank-1 edit toward a learned target, with the identity map as its closed-gate limit. At GPT-2 small and medium sizes, every reported DDL variant reaches a lower final validation loss than the baseline in single runs with equal token budgets. The scalar-state gains are small. The expanded-state gains are larger, but they combine residual capacity, compression, and rewriting, and they cost memory and throughput. Attributing them to the rewrite requires a matched write-only control; testing their significance requires multiple seeds; and an equal-compute comparison requires compute-matched training curves. We have not yet measured which gate regimes trained models use or how much each layer reduces its discrepancy.

AI use statement

We used generative AI tools to help implement the experiment code. We did not use them to develop the theory, formulate or prove the mathematical results, propose hypotheses, design the experiments, or interpret the results; generating or cleaning datasets, qualitative data analysis, and translation do not apply to this work. We also used generative AI tools to edit the paper for readability and migrate it to the ICLR template. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility statement

The supplementary material contains the model code for every reported variant. Appendix A maps each variant to its model file, and the reported runs use each file's default configuration. Section 3 specifies the direction, value, and gate generators and the TC and CC compressors, Appendix A gives their configuration keys, and Table 2 lists the architecture hyperparameters. Section 4 gives the data, token budget, batch size, sequence length, optimizer, learning-rate schedule, hardware, and evaluation benchmarks, which are run with `lm-evaluation-harness`. The analysis in Section 2, including the proof of Proposition 2.1, is self-contained.

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*, pp. 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, pp. 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*, pp. 6887–6900, 2020.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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*, pp. 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*, pp. 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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 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 Le Bras, 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*, pp. 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, pp. 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.

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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

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)*, pp. 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.

A Implementation Details

The reported variants are implemented in the following files under `model/`. Each file also has a Triton implementation with the suffix `-accelerated`.

Variant	File	Initial state	Compressor
DDL ($d_v = 1$)	`DDL-vdim1-gpt-mha-rope-TC.py`	–	–
DDL-TC w/o EC	`DDL-gpt-mha-rope-TC.py`	repeated embedding	TC
DDL-TC	`DDL-gpt-mha-rope-TC-EC.py`	EC	TC
DDL-CC w/o EC	`DDL-gpt-mha-rope-CC.py`	repeated embedding	CC
DDL-CC	`DDL-gpt-mha-rope-CC-EC.py`	EC	CC

The reported runs use each file's default configuration. The defaults correspond to the settings in Sections 3, 3.1, and 3.2 as follows:

Configuration key	Setting
`ddl_v_sigmoid`, `ddl_v_sigmoid_scale`	sigmoid value branch with scale 4
`ddl_beta_init`	initial gate $\beta_l = 1$
`ddl_k_eps`	$\epsilon_k = 10^{-5}$
`input_embed_shortconv_kernel_size`	EC kernel size $k_e = 4$
`ddl_state_shortconv_kernel_size`	compressor kernel size $k = 4$
`ddl_state_read_init`	initial read vector $\mathbf{r} = 1/d_v$

The code also provides a two-layer $\tanh$ gate, which the reported runs do not use.