DELTA ATTENTION RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention Residuals (AttnRes) replace the standard residual connection with learned softmax attention over previous sublayer outputs, and the routed mixture *replaces* the residual stream as the next sublayer's input. We find that this replacement limits AttnRes at scale: it improves on standard residuals at 220M but is 6.9% worse in perplexity at 1044M and 6.6% worse at 7.6B, and the variant without any residual stream degrades most. We propose **Delta Attention Residuals**, whose core change is the update rule: they keep the residual stream and *add* a softmax-routed mixture of *deltas* ($\mathbf{v}_i = \mathbf{h}_{i+1} - \mathbf{h}_i$) to it. Under an additive update, the natural alternative source, cumulative hidden states, is highly redundant (adjacent cosine 0.93–0.97, R^2 up to 0.99 in deep layers) and gives higher perplexity than deltas on two architecture families. Across 220M–7.6B, Delta Attention Residuals reduce validation perplexity by 1.7–8.2% relative to standard residuals, and a zero-initialized gate on the added term supports fine-tuning from pretrained checkpoints. Code is in the supplementary material.

1 INTRODUCTION

Residual connections (He et al., 2016) in transformers (Vaswani et al., 2017) accumulate sublayer outputs through the update $\mathbf{h}_l = \mathbf{h}_{l-1} + f_l(\mathbf{h}_{l-1})$. This provides direct paths for gradient propagation (Veit et al., 2016), but weights all preceding outputs equally. Attention Residuals (Kimi Team, 2026) replace this fixed aggregation with learned softmax attention over the token embedding and the outputs of preceding sublayers (Full AttnRes), or over per-block sums of those outputs (Block AttnRes), allowing the model to weight contributions from different depths. The routed mixture replaces the residual stream as the input to the next sublayer.

AttnRes thus also changes the update: the routed mixture *replaces* the residual stream, so the next sublayer no longer receives the running sum as an identity path. In our reimplementation this costs more as models grow: AttnRes improves on standard residuals at 220M but is 6.9% worse at 1044M, Full AttnRes, which keeps no stream across sublayers, is 12.3% worse (§3.2), and AttnRes is 6.6% worse at 7.6B (§3.3).

We therefore keep the stream and make routing *additive*, which raises the question of what to add. Cumulative hidden states $\mathbf{h}_i$ already contain the stream up to depth i, and with depth they become nearly linear functions of one another (Razzhigaev et al., 2024): on trained checkpoints, the newest one is predictable from earlier ones with R^2 up to 0.99 in deep layers, and adjacent ones reach cosine 0.97 (§4.1). The delta $\mathbf{v}_i = \mathbf{h}_{i+1} - \mathbf{h}_i$ instead isolates the change introduced by sublayer i: adjacent deltas stay near-orthogonal at every depth (cosine $|\cdot| \leq 0.06$) (Elhage et al., 2021), on both Qwen3- and LLaMA-style models.

We propose **Delta Attention Residuals**, which add a softmax-routed mixture of per-sublayer or block-level deltas to the stream, $\mathbf{h} = \tilde{\mathbf{h}} + \sum \alpha_i \mathbf{v}_i$. Because the stream is never replaced, a zero-initialized gate on the added term leaves a pretrained model unchanged at initialization.

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.

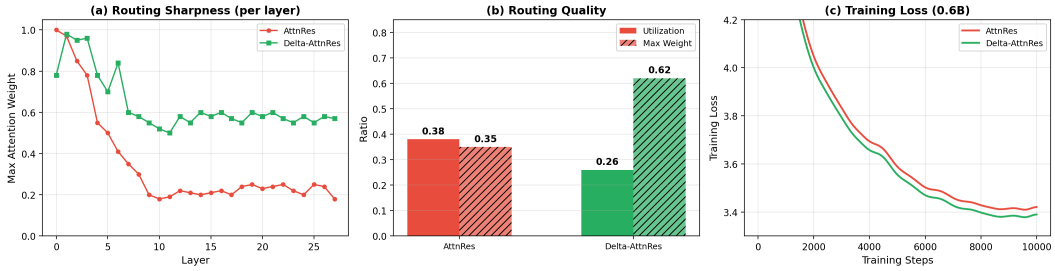

Figure 1: **Routing behavior of replacement vs. additive routing** (Qwen3-0.6B, L=28). (a) Maximum routing weight by layer: AttnRes has lower max weight in deep layers ($\sim$0.2), Delta Block higher ($\sim$0.6); max weight is unnormalized; N_{eff}/N in Appendix E.1. (b) Average max weight (0.62 vs. 0.35). (c) Training loss: Delta Block (green) has lower loss than AttnRes (red).

Our contributions are:

1. We show that the **replacement update** of Attention Residuals limits them at scale, and propose **Delta Attention Residuals**, which keep the residual stream and *add* a routed mixture of deltas, improving validation perplexity over baseline and AttnRes at 220M–7.6B (§3).
2. We show that under the additive update, **cumulative sources are a poor choice**: they are highly redundant and give higher perplexity than deltas on two architecture families (§4, Appendix E).
3. The additive form admits a **zero-initialized gate** for converting pretrained transformers; fine-tuned Delta Block improves on its pretrained checkpoint across 8 benchmarks (§3.5).

2 Method: Delta Attention Residuals

2.1 Preliminaries: Attention Residuals

A Pre-Norm transformer (Vaswani et al., 2017; Xiong et al., 2020) updates the hidden state as $\mathbf{h}_{i+1} = \mathbf{h}_i + \mathbf{v}_i$, where $\mathbf{v}_i := f_i(\mathbf{h}_i)$ is defined as the output of sublayer i—here a sublayer can be either an attention or MLP layer. Equivalently, $\mathbf{v}_i = \mathbf{h}_{i+1} - \mathbf{h}_i$ also measures the change introduced at depth i. Standard residuals accumulate all sublayer outputs with fixed unit coefficients. Attention Residuals (Kimi Team, 2026) replace this with a learned weighted sum:

$$\hat{\mathbf{h}}_l = \sum_{i=0}^{l-1} \alpha_{i\to l} \cdot \mathbf{s}_i, \quad \alpha_{i\to l} = \frac{\exp(\mathbf{w}_l^\top \mathrm{RMSNorm}(\mathbf{s}_i))}{\sum_j \exp(\mathbf{w}_l^\top \mathrm{RMSNorm}(\mathbf{s}_j))} \tag{1}$$

where $\mathbf{s}_i$ are the *source* representations from preceding layers $i \in [0, l)$, $\mathbf{w}_l \in \mathbb{R}^d$ is a learned query, and $\hat{\mathbf{h}}_l$ feeds into the next sublayer.

2.2 What to Add: Cumulative vs. Delta Sources

We consider two source representations for $\mathbf{s}_i$ under the additive update of §2.3:

Cumulative sources. $\mathbf{s}_i = \mathbf{h}_i = \mathbf{h}_0 + \sum_{j=1}^{i} \mathbf{v}_j$—the full hidden state after sublayer i. This is not the source choice of AttnRes, which routes over sublayer outputs (Full) or intra-block sums of them (Block), but it is the natural choice when the stream is kept, and we train it as a control. Under an additive update it has two drawbacks: each $\mathbf{h}_i$ already contains the stream up to depth i, so the added mixture largely re-adds what the stream carries, and adjacent states share a growing common prefix, so the router chooses among near-duplicates. We measure this redundancy in §4.1. LIMe (Gerasimov et al., 2026) also routes over earlier states.

Delta sources. $\mathbf{s}_i = \mathbf{v}_i$—the per-sublayer output defined in §2. At the finest granularity, each attention output and each MLP output is a separate source (**Delta AttnRes**, $2L$ sources for L layers). In our measurements, adjacent deltas have substantially lower cosine similarity than cumulative sources. We also consider block-level grouping: a block delta $\Delta_b = \mathbf{h}_{b+1} - \mathbf{h}_b$ aggregates multiple

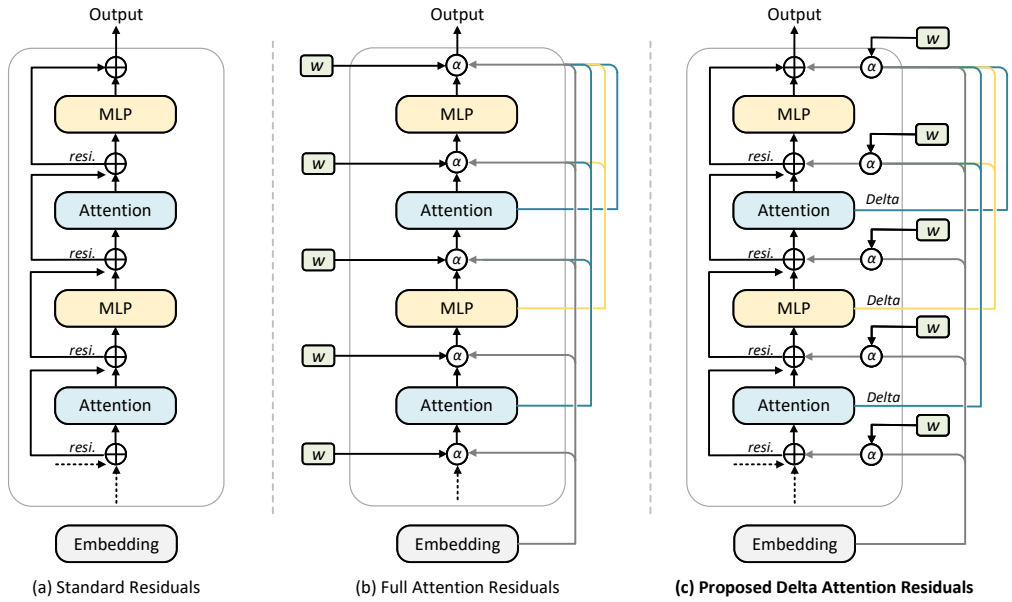

Figure 2: **Architecture comparison.** (a) Standard Residuals: uniform additive accumulation with fixed unit coefficients. (b) Attention Residuals (Kimi Team, 2026): learned softmax attention (weights $\mathbf{w}$, aggregation a) over the embedding and preceding sublayer outputs (or per-block sums of them); the routed mixture replaces the residual stream as the next sublayer's input. (c) **Delta Attention Residuals** (ours): the residual stream is kept, and a routed mixture of per-sublayer delta outputs, which remain near-orthogonal in our measurements, is added to it.

sublayer outputs into one source (**Delta Block**, $B{+}1$ sources for B blocks), trading granularity for efficiency. See Appendix C for a schematic and Appendix C for the source construction used in the experiments.

2.3 Delta Attention Residuals

Delta Attention Residuals change the update rule: instead of replacing the stream (Eq. 1), they use *additive routing*, adding an attention-weighted sum of delta sources to the current residual stream:

$$\hat{\mathbf{h}}_l = \tilde{\mathbf{h}}_l + \sum_i \alpha_{i\to l} \cdot \mathbf{v}_i \tag{2}$$

where $\mathbf{v}_i$ are per-sublayer outputs, $\tilde{\mathbf{h}}_l$ is the current residual stream, and $\alpha_{i\to l} = \mathrm{softmax}(\mathbf{w}_l^\top \mathrm{RMSNorm}(\mathbf{v}_i))$. This formulation has two advantages:

1. **Residual preservation.** The residual stream $\tilde{\mathbf{h}}_l$ is preserved by default; routing adds information rather than replacing it.
2. **Safe initialization.** When converting a pretrained model, the routed term enters through a zero-initialized scalar gate, $\hat{\mathbf{h}}_l = \tilde{\mathbf{h}}_l + g_l \sum \alpha_i \mathbf{v}_i$ with $g_l{=}0$, so the gated routing update leaves the residual stream unchanged at initialization ($\hat{\mathbf{h}}_l = \tilde{\mathbf{h}}_l$; §3.5).

Figure 3 gives pseudocode for additive routing and source construction; a comparison with the original AttnRes is in Appendix B.

```python
def depth_route(sources: list[Tensor], residual: Tensor,
                proj: Linear, norm: RMSNorm) -> Tensor:
    """Softmax attention over depth sources, added to residual."""
    V = torch.stack(sources)                          # [N, B, T, D]
    K = norm(V)
    logits = einsum('d, n b t d -> n b t', proj.weight.squeeze(), K)
    return residual + einsum('n b t, n b t d -> b t d', logits.softmax
    (0), V)

def forward(self, deltas, hidden_states):
    # --- Attention sublayer ---
    h = depth_route(deltas, hidden_states, self.attn_proj, self.
    attn_norm)
    attn_out = self.attn(norm(h))
    if self.mode == "Delta Block" and self.is_block_boundary:
        if not deltas:
            deltas.append(hidden_states)              # source 0:
    token embedding
        else:
            deltas.append(hidden_states - deltas[-1])  # difference
    from last source
    hidden_states = hidden_states + attn_out
    if self.mode == "Delta AttnRes":
        deltas.append(attn_out)                       # store attention
    delta

    # --- MLP sublayer ---
    h = depth_route(deltas, hidden_states, self.mlp_proj, self.
    mlp_norm)
    mlp_out = self.mlp(norm(h))
    hidden_states = hidden_states + mlp_out
    if self.mode == "Delta AttnRes":
        deltas.append(mlp_out)                        # store MLP delta

    return deltas, hidden_states
```

Figure 3: **Delta Attention Residuals pseudocode.** The routing operator adds a softmax-weighted mixture of delta sources to the residual stream. **Delta Block** stores the token embedding as source 0 and, at each later boundary, the difference between the current state and the previously stored source; **Delta AttnRes** appends every sublayer output directly. Appendix C gives the mathematical form of the block-source recurrence.

3 Experiments

3.1 Experimental Setup

We train Qwen3-architecture (Qwen, 2025) models from scratch on FineWeb-Edu (Penedo et al., 2024):

- **Scales**: 220M (d=768, L=12), 533M (d=1024, L=24), 1044M (d=1280, L=36), plus standard Qwen3-0.6B (d=1024, L=28) and Qwen3-8B (d=4096, L=36) configurations
- **Training**: AdamW (Loshchilov & Hutter, 2019) (β_1=0.9, β_2=0.95, weight decay 0.1), cosine learning-rate decay with 500-step warmup, learning rate 6×10^{-4} (220M–1044M) or 3×10^{-4} (8B)
- **Budget**: 10K steps, effective batch size 32 (220M–1044M) or 64 (0.6B, 8B), sequence length 1024 (220M–1044M) or 2048 (0.6B, 8B)
- **Hardware**: 8× NVIDIA H100 80GB, BF16 mixed precision, `torch.compile`

All models are compiled with `torch.compile` (default mode) before DDP wrapping. Throughput (Tok/s) is measured as total tokens across all GPUs per wall-clock second at steady state. Peak memory (Mem) is the maximum GPU memory allocated to tensors per device during training (batch size 4 per GPU, sequence length 1024), measured with `torch.cuda.max_memory_allocated`.

Table 1: Results across three model scales (10K steps, FineWeb-Edu). All methods share identical architecture, data, and hyperparameters per scale; only the depth-mixing mechanism differs. Best per scale in **bold**. Tok/s measured on 8×H100 (BF16, `torch.compile`); Mem is peak per-device GPU memory during training (batch size 4 per GPU, sequence length 1024).

Scale	Method	Train Loss ↓	Val Loss ↓	Val PPL ↓	Tok/s ↑	Mem (GB)
220M d=768 L=12	Baseline	3.613	3.656	38.71	289k	5.6
	AttnRes	3.586	3.622	37.39	294k	6.2
	Full AttnRes	3.577	3.619	37.30	285k	6.8
	Delta Block	3.577	3.613	37.08	306k	5.9
	Delta AttnRes	**3.563**	**3.606**	**36.83**	246k	7.5
533M d=1024 L=24	Baseline	3.428	3.466	32.00	243k	13.9
	AttnRes	3.423	3.458	31.75	174k	17.5
	Full AttnRes	3.422	3.456	31.68	103k	24.8
	Delta Block	3.405	3.439	31.16	198k	16.1
	Delta AttnRes	**3.401**	**3.436**	**31.05**	80k	33.6
1044M d=1280 L=36	Baseline	3.360	3.391	29.70	108k	22.5
	AttnRes	3.428	3.458	31.76	71k	31.6
	Full AttnRes	3.474	3.508	33.36	49k	52.0
	Delta Block	**3.338**	3.374	29.19	86k	28.4
	Delta AttnRes	3.339	**3.372**	**29.13**	34k	77.7

We evaluate five configurations: **Baseline** (standard residual), **AttnRes** (Block AttnRes: replacement routing over the embedding and per-block sums of sublayer outputs, following Kimi Team (2026)), **Full AttnRes** (AttnRes with N=L, i.e. each layer as its own block), **Delta AttnRes** (per-sublayer delta sources with additive routing), and **Delta Block** (delta sources with block-level grouping and additive routing). Since the original Attention Residuals implementation is not publicly available, AttnRes and Full AttnRes are our faithful reimplementation based on the description in Kimi Team (2026) (see Appendix B for details). All AttnRes variants use the same query initialization and identical hyperparameters per scale.

3.2 From-Scratch Training Results

Validation perplexity. Delta AttnRes achieves the lowest validation PPL at all three scales: 36.83 at 220M, 31.05 at 533M, and 29.13 at 1044M. Delta Block reaches 37.08, 31.16, and 29.19, respectively, within 0.7% of Delta AttnRes at each scale. Both delta methods improve over baseline; the Delta AttnRes improvement narrows from 4.9% at 220M to 3.0% at 533M and 1.9% at 1044M.

Replacement routing degrades at scale. AttnRes and Full AttnRes both use replacement routing: the routed mixture, rather than a persistent residual stream, is the next sublayer's input. Full AttnRes keeps no residual stream across sublayers, while AttnRes keeps only an intra-block partial sum that restarts at each of its 4 block boundaries. At 220M, AttnRes (37.39) and Full AttnRes (37.30) improve over baseline (38.71). At 1044M (L=36), their PPL is 31.76 and 33.36, respectively, exceeding baseline's 29.70 by 6.9% and 12.3%. The larger degradation for the configuration without any persistent residual stream is consistent with losing the identity path (§4.2).

Source and update choices. Full AttnRes already routes over per-sublayer outputs, the same sources as Delta AttnRes, and Block AttnRes routes over intra-block sums of them. The principal difference is therefore the update rule: delta routing is additive ($\mathbf{h} = \tilde{\mathbf{h}} + \sum \alpha_i \mathbf{v}_i$) rather than replacement ($\mathbf{h} = \sum \alpha_i \mathbf{s}_i$), so it keeps the residual stream and adds the routed result, with no block-level restart (§2). Delta Block additionally differs from Block AttnRes in how block sources are constructed (Appendix C). At 1044M, AttnRes is 6.9% worse than baseline, while Delta Block is 1.7% better. The controlled ablation in §4.2 separates the effects of these two choices.

Throughput and memory. Per-sublayer Delta AttnRes stores $2L$ sources, reducing throughput by 69% and using $3.5\times$ the memory of baseline at 1044M (34k tok/s, 77.7 GB vs. 108k, 22.5 GB). Delta

Block reduces this cost through block-level grouping: at 1044M it runs at 86k tok/s with 28.4 GB (20% throughput reduction, 26% memory increase), with a PPL gap of approximately 0.2% relative to Delta AttnRes (29.19 vs. 29.13).

3.3 Scaling From-Scratch Training

The results in Table 1 use custom model configurations. We first verify that the same trends hold on an existing architecture, then scale up to 8B parameters.

Existing architecture: Qwen3-0.6B. We train from scratch using the standard Qwen3-0.6B architecture (d=1024, L=28, 508M params) with per-layer routing (N=L=28). Delta Block improves over baseline by 2.4% (31.45 vs. 32.22 PPL), while AttnRes slightly degrades (32.38), consistent with its degradation at 1044M and 8B. Figure 4 shows higher max weight for Delta Block ($\sim$0.6) throughout depth than for AttnRes ($\sim$0.2) in deep layers (max weight is unnormalized; N_{eff}/N in Appendix E.1).

Table 2: From-scratch training at Qwen3-0.6B scale (L=28, N=28, 10K steps, learning rate 6×10^{-4}).

Method	Val PPL ↓	Tok/s ↑	Mem (GB)
Baseline	32.22	173k	13.8
AttnRes (N=28)	32.38	111k	28.4
Delta Block (N=28)	**31.45**	117k	25.2

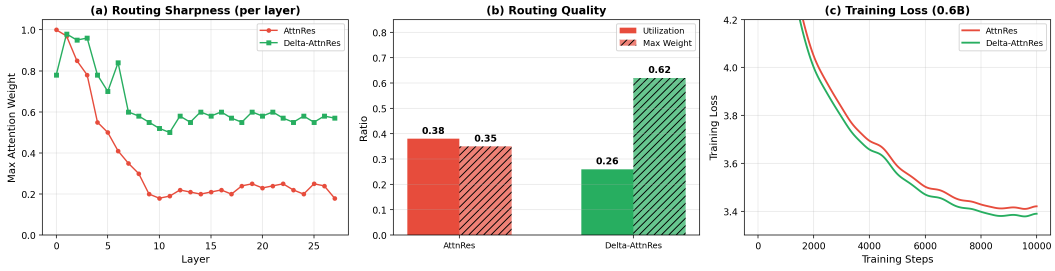

Figure 4: **Routing analysis at Qwen3-0.6B scale** (L=28, N=28). (a) Delta Block has higher max weight ($\sim$0.6–1.0) than AttnRes ($\sim$0.2) in deep layers; max weight is unnormalized; N_{eff}/N in Appendix E.1. (b) Delta Block achieves 1.8× higher average max weight (0.62 vs. 0.35). (c) Training loss: Delta Block (green) consistently outperforms AttnRes (red).

Scaling up: 8B parameters. We next scale to a Qwen3-8B-sized model (d=4096, L=36, 7.57B params) trained from scratch on FineWeb-Edu for 10K steps with FSDP and gradient checkpointing on 8×H100 (learning rate 3×10^{-4}, sequence length 2048, effective batch 64).

Table 3: **Scaling to 8B** (10K steps, FineWeb-Edu, sequence length 2048, batch 64, 8×H100 FSDP). Delta Block achieves the best PPL while AttnRes degrades below baseline, consistent with the 1044M trend. Tok/s and Mem measured on 8×H100 (BF16, `torch.compile`, gradient checkpointing). Best in **bold**.

Scale	Method	Train Loss ↓	Val Loss ↓	Val PPL ↓	Tok/s ↑	Mem (GB)
7.57B	Baseline	2.918	2.858	17.43	21.4k	41.6
d=4096	AttnRes	2.984	2.922	18.58	12.5k	44.0
L=36	**Delta Block**	**2.828**	**2.773**	**16.00**	14.0k	42.7

Validation perplexity at 8B. Delta Block reaches 16.00 PPL, an 8.2% reduction from baseline's 17.43. AttnRes reaches 18.58 (6.6% above baseline), consistent with the degradation observed at 1044M (6.9%, Table 1).

Computational overhead. Delta Block adds 589.8K routing parameters (0.008% of 7.57B). It runs at 14.0k tok/s vs. baseline's 21.4k (35% throughput reduction) and uses 42.7 GB vs. 41.6 GB (3% memory increase). Compared with AttnRes, it has higher throughput (14.0k vs. 12.5k tok/s) and lower memory use (42.7 vs. 44.0 GB).

3.4 Ablation: Effect of Block Size

We ablate the number of blocks for Delta Block at 220M (L=12) and 533M (L=24). B=4 is the default used in Table 1.

Table 4: Delta Block block size ablation (10K steps). Tok/s on 8×H100.

Method	B	Sources	Val PPL ↓	Tok/s ↑	Mem (GB)
220M (L=12)					
Baseline	—	—	38.71	289k	5.6
Delta Block	1	2	37.44	396k	8.7
Delta Block	2	3	37.08	318k	8.8
Delta Block	4	5	36.98	343k	9.1
Delta Block	6	7	**36.92**	315k	9.3
Delta Block	12	13	37.34	318k	10.2
Delta AttnRes	—	25	**36.75**	212k	12.4
533M (L=24)					
Baseline	—	—	32.00	243k	13.9
Delta Block	2	3	31.23	217k	14.7
Delta Block	4	5	31.27	160k	15.4
Delta Block	6	7	**31.19**	194k	16.1
Delta Block	12	13	31.22	171k	18.3
Delta Block	24	25	31.18	137k	23.5
Delta AttnRes	—	49	**31.05**	80k	33.6

220M: lowest block-level PPL at B=6. Delta Block improves from B=1 (37.44) to B=6 (36.92), then degrades at B=12 (37.34). Full per-sublayer Delta AttnRes (36.75) achieves the best PPL at this scale.

533M: limited sensitivity to block count. At 533M, Delta Block PPL varies from 31.18 to 31.27 as the block count increases from B=2 to B=24. Using fewer blocks therefore has little effect on perplexity in this setting.

Choice of block count. Delta AttnRes achieves the best PPL at both scales but uses $2L$ sources. We use Delta Block with B=4 as the default: at 220M, its PPL is 0.6% higher than Delta AttnRes while throughput is 1.6× higher.

3.5 Fine-Tuning Pretrained Models into Delta Attention Residuals

Prior work (Pagliardini et al., 2024) found diminishing gains when cross-layer weights were introduced only after part of pretraining, suggesting that early training affects their subsequent use. We evaluate whether additive delta routing with zero-initialized gates (§2, advantage 2) can improve pretrained checkpoints through fine-tuning.

Setup. We fine-tune Qwen3-0.6B (Qwen, 2025) on FineWeb-Edu (Penedo et al., 2024) for 20K steps (warmup 500, cosine decay, batch size 32, 4×H100). Inspired by LoRA (Hu et al., 2022), we use a dual learning rate for Delta Block: 5×10^{-5} for the pretrained transformer and 5×10^{-3} for the AttnRes parameters, allowing the lightweight routing module to train faster while preserving pretrained knowledge. Baseline and AttnRes use a uniform learning rate of 5×10^{-5}. We compare: (1) Baseline (standard fine-tuning), (2) AttnRes (Kimi Team, 2026), and (3) Delta Block with an additional learnable routing source and a zero-initialized gate g_l (ours). Each attention and MLP routing site has its own additional source vector, initialized to zero and included with the

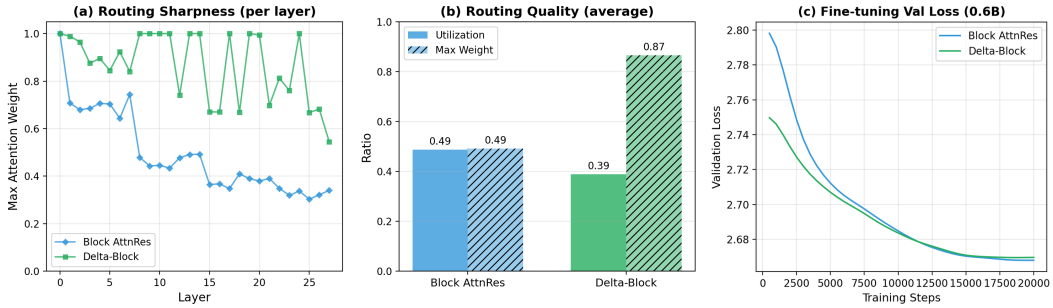

Figure 5: **Routing analysis** (Qwen3-0.6B fine-tuned on FineWeb-Edu). (a) Per-layer max weight: Delta Block has higher max weight ($\sim$0.87) throughout depth, while AttnRes has lower max weight (0.7 to 0.3); max weight is unnormalized; N_{eff}/N in Appendix E.1. (b) Average max weight: Delta Block achieves $1.8\times$ higher max weight (0.87 vs. 0.49). (c) Validation loss: AttnRes (blue) starts higher due to initialization disruption and converges slower; Delta Block (green) outperforms baseline.

delta sources in the routing softmax. We evaluate on 8 standard benchmarks using lm-evaluation-harness (Gao et al., 2023) (0-shot): HellaSwag (Zellers et al., 2019), ARC-Easy/Challenge (Clark et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), BoolQ (Clark et al., 2019), MMLU (Hendrycks et al., 2021), and LAMBADA (Paperno et al., 2016).

Table 5: **Fine-tuning Qwen3-0.6B on FineWeb-Edu.** Downstream accuracy (0-shot) on 8 benchmarks. Delta Block outperforms baseline (55.6% vs. 55.0%), while AttnRes lags behind (54.1%).

Method	Hella	ARC-E	ARC-C	PIQA	Wino	BoolQ	MMLU	LAMB	Avg
Pretrained	47.3	56.2	33.9	67.4	55.9	63.8	40.3	40.0	50.6
Baseline FT	50.1	61.5	36.7	69.2	57.0	70.3	45.2	50.0	55.0
AttnRes	49.4	60.2	35.3	68.8	57.9	70.3	42.5	48.4	54.1
Delta Block	50.0	**66.5**	**37.3**	**69.4**	56.7	70.3	44.8	49.8	**55.6**

Results. Delta Block achieves 55.6% average accuracy, outperforming baseline (55.0%) and AttnRes (54.1%), with the highest ARC-Easy (66.5) and ARC-Challenge (37.3) scores.

Initialization and training loss. In Figure 5, AttnRes starts with a loss increase from 2.8 to 3.96 and takes $\sim$2000 steps to recover. Its replacement routing ($\mathbf{h} = \sum \alpha_i \mathbf{s}_i$) substitutes the routed mixture for the pretrained residual stream. Delta Block starts from the pretrained loss: its additive formulation ($\mathbf{h} = \tilde{\mathbf{h}} + \sum \alpha_i \mathbf{v}_i$), combined with a zero-initialized gate, preserves the pretrained computation at initialization (§2, advantage 2). AttnRes also retains a downstream accuracy gap after fine-tuning (Table 5).

4 Analysis

4.1 Source Redundancy Across Depth

Each cumulative state $\mathbf{h}_i = \mathbf{h}_0 + \sum_{j=1}^{i} \mathbf{v}_j$ shares preceding sublayer outputs with its neighbors. On trained checkpoints evaluated over held-out data (Appendix E), the predictability of the newest cumulative source from earlier ones increases with depth: R^2 rises from 0.46 to 0.97 at 220M and from 0.45 to 0.99 at 533M. Adjacent-source cosine reaches 0.93/0.97 at relative depth 0.9. Delta sources remain near-orthogonal across the measured depths (cosine $|\cdot| \leq 0.06$), with deep-layer R^2 of 0.22/0.28. The same contrast holds on the LLaMA family (adjacent cosine $\approx$0.9 cumulative vs. $\approx$0 delta; Appendix E).

Routing concentration also differs between configurations. In the deep third of the model, the effective source fraction N_{eff}/N is 0.33 (220M) and 0.18 (533M) for cumulative+replacement, compared with 0.19 and 0.09 for delta+additive. Normalized entropy gives the same ordering ($H/\ln N$ of 0.76

Table 6: 2×2 **ablation**: sources × update rule (validation PPL ↓, 10K steps, per-sublayer sources, single seed; baselines 38.71/32.00 Qwen, 38.49/33.65 LLaMA). Delta+additive is best in all four settings; protocol in Appendix D.

Sources	Update	Qwen 220M	Qwen 533M	LLaMA 220M	LLaMA 533M
cumulative	replacement	37.05	33.17	36.61	33.20
cumulative	additive	38.10	31.57	37.54	32.00
delta	replacement	37.90	31.66	38.10	32.67
delta	additive	**36.82**	**31.03**	**36.48**	**31.24**

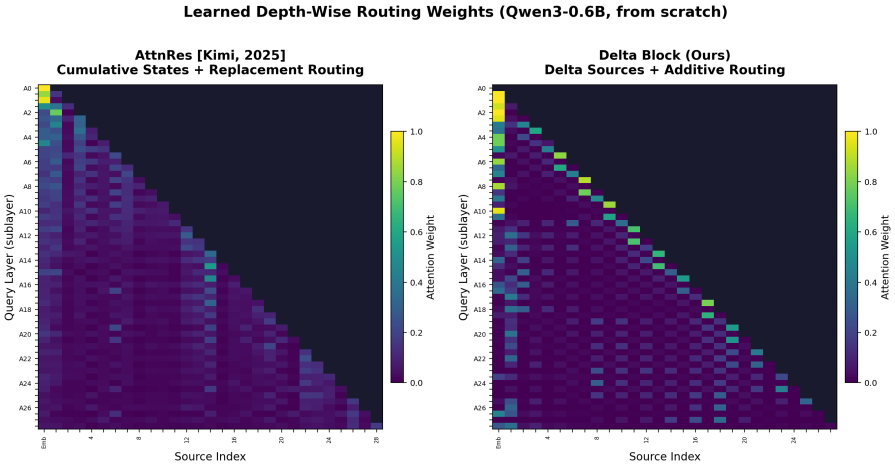

Figure 6: **Learned routing weights** (Qwen3-0.6B, from scratch). **Left**: AttnRes (Kimi Team, 2026) with block-sum sources and replacement routing assigns more diffuse weights in deep layers. **Right**: Delta Block (ours) with delta sources and additive routing concentrates weight on specific early outputs.

vs. 0.47 at 220M; Appendix E.1). Cumulative routing is therefore nonuniform despite a maximum weight near 0.2. This statistic alone does not establish routing collapse; source similarity and weight concentration describe different properties of the routing operation.

4.2 Why the Additive Update Matters

AttnRes replaces the hidden state ($\mathbf{h} = \sum \alpha_i \mathbf{s}_i$). Its block restarts do not erase history, since completed blocks stay available to the router; what is lost is the identity path. Full AttnRes is itself replacement routing over the same delta sources as Delta AttnRes, differing only in the update rule; it keeps no persistent stream and degrades most at 1044M (§3.2). With the additive update fixed, delta sources beat cumulative sources in all four settings of Table 6 (0.54–1.28 PPL), and delta+additive is the best cell throughout.

4.3 Learned Routing Patterns

Figure 6 visualizes the learned routing weights at Qwen3-0.6B scale (L=28). AttnRes (left) assigns increasingly diffuse weights in deeper layers. In Delta Block (right), deep layers assign >50% weight to specific early outputs, and high weight to the token embedding, adding embedding information to the current residual stream.

5 Conclusion

Delta Attention Residuals add a routed mixture of deltas to the residual stream instead of replacing the stream as Attention Residuals do. Replacement routing falls behind standard residuals at 1044M and 7.6B, whereas the additive variants improve at every tested scale, and deltas are a better source than redundant cumulative states. At 7.6B, Delta Block reduces perplexity by 8.2%, adds 0.008% parameters, and runs faster than AttnRes.

AI USE STATEMENT

In this work, we used generative AI tools to help *implement methods*: an AI coding assistant (Claude) helped write and refactor the model, training, fine-tuning, and evaluation code. All AI-assisted code was reviewed and tested by the authors, and the main results (Tables 1–5) are mapped to logged training runs in the supplementary material. We have not used generative AI tools to propose or refine hypotheses, to design the method or the experiments, or to interpret results; the remaining required-disclosure tasks (synthetic data generation, theoretical frameworks, mathematical claims or proofs, translation, dataset cleaning, and qualitative data analysis) are not applicable to this work. Additionally, we used generative AI tools to edit software code, to reformat the manuscript into the conference template, to edit the text for readability, and to draft and condense parts of the text (including appendix analyses written up from our measurements), which we checked against the code and logs. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 2 describes the routing method, and Figure 3 provides pseudocode. Section 3 specifies model configurations, optimizer settings, hardware, and training budgets for every experiment, and Appendices B and C detail the differences from the original Attention Residuals and the block-level variant. All experiments use the publicly available FineWeb-Edu corpus and the Qwen3 architecture, plus a LLaMA-architecture replication (Appendix F) that uses the same source representations and update rules. Code for the Qwen3 experiments is included (anonymized) in the supplementary material.

REFERENCES

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. *UAI*, 2021.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*, 2020.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. DoLa: Decoding by contrasting layers improves factuality in large language models. In *ICLR*, 2024.

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

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.

Nelson Elhage, Neel Nanda, Catherine Olsson, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2023. URL `https://zenodo.org/records/10256836`.

Gleb Gerasimov, Yaroslav Aksenov, Nikita Balagansky, Viacheslav Sinii, and Daniil Gavrilov. You do not fully utilize transformer's representation capacity. In *EMNLP*, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. Realformer: Transformer likes residual attention. *ACL Findings*, 2021.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *ICLR*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *ICLR*, 2022.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Kimi Team. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *ACL*, 2023.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. Tuning language models by proxy. In *COLM*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *arXiv preprint arXiv:2402.02622*, 2024.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL*, 2016.

Guilherme Penedo, Hynek Kydlíček, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Qwen. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Anton Razzhigaev, Matvey Mikhalchuk, Elizaveta Goncharova, Nikolai Gerasimenko, Ivan Oseledets, Denis Dimitrov, and Andrey Kuznetsov. Your transformer is secretly linear. In *ACL*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *AAAI*, 2020.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V Le, Geoffrey E Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ICLR*, 2017.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *NeurIPS*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *NeurIPS*, 2016.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. *ICML*, 2025.

Zhenda Xie et al. mHC: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *ICML*, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL*, 2019.

Yifan Zhang. Residual stream duality in modern transformer architectures. *arXiv preprint arXiv:2603.16039*, 2026.

Yifan Zhang, Yifeng Liu, Mengdi Wang, and Quanquan Gu. Deep delta learning. *arXiv preprint arXiv:2601.00417*, 2026.

Defa Zhu et al. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024.

A Related Work

Cross-Layer Connections. DenseNet (Huang et al., 2017) concatenates all previous outputs ($O(L^2d)$ memory). DenseFormer (Pagliardini et al., 2024) uses static Depth-Weighted Average weights. Hyper-Connections (Zhu et al., 2024) widen the residual stream to n channels; mHC (Xie et al., 2025) constrains mixing to the doubly stochastic manifold. MUDDFormer (Xiao et al., 2025) generates dynamic, per-stream weights via MLPs, showing 2.8B matches 6.9B Pythia. Realformer (He et al., 2021) passes residual attention scores across layers via additive skip connections on the attention logit matrix. Attention Residuals (Kimi Team, 2026) use softmax attention over depth, scaling to a 48B-parameter MoE (3B activated). LIMe (Gerasimov et al., 2026) instead routes inside attention: each head learns per-layer weights over earlier layers' key-value representations, motivated by representation collapse when a single hidden state must carry the whole context. Attention Residuals replace the residual stream with the routed mixture; we instead add the routed mixture to the stream, and we route over per-sublayer deltas, whereas LIMe routes over projections of earlier hidden states.

Residual Stream Analysis. Veit et al. (2016) view residual networks as ensembles; Elhage et al. (2021) formalize the residual stream. DeepNet (Wang et al., 2022) rescales residual connections with a depth-dependent factor to stabilize 1000-layer transformers. Residual stream duality (Zhang, 2026) shows depth attention is dual to sequence-axis sliding-window attention. Our source redundancy analysis complements these views.

Contrastive and Delta Methods. Several methods use differences between model outputs or representations. Contrastive Decoding (Li et al., 2023) subtracts amateur logits from expert logits to isolate scale-dependent knowledge. DoLa (Chuang et al., 2024) contrasts the vocabulary logits of later versus earlier layers within a single model, exploiting the fact that factual knowledge is localized to particular layers. Proxy-Tuning (Liu et al., 2024) transfers fine-tuning residuals across model scales at decode time. Our work applies differencing along the depth axis within a single forward pass.

Gating and Routing. Highway Networks (Srivastava et al., 2015) introduced learned gating of skip connections. GLU (Shazeer, 2020) gates and MoE (Shazeer et al., 2017; Fedus et al., 2022) routes along the width dimension, while ReZero (Bachlechner et al., 2021) learns a scalar gate on each residual branch; Mixture of Depths (Raposo et al., 2024) routes tokens along depth via per-token top-k decisions. We route *source representations* along depth. Deep Delta Learning (Zhang et al., 2026) modulates the identity shortcut with a gated rank-1 (Householder-like) operator; we use attention-based selection.

B Comparison with Original Attention Residuals

We reproduce the original AttnRes pseudocode from Kimi Team (2026) and analyze the key implementation differences with Delta Attention Residuals.

Table 7: Implementation differences between AttnRes and Delta Attention Residuals.

Property	**AttnRes**	**Delta AttnRes**	**Delta Block**
Sources	$\mathbf{v}_i$ (Full) / block sums (Block)	sublayer $\mathbf{v}_i$	block Δ_b
Routing	replacement	additive	additive
	$h = \sum \alpha_i \mathbf{s}_i$	$h = \tilde{\mathbf{h}} + \sum \alpha_i \mathbf{v}_i$	$h = \tilde{\mathbf{h}} + \sum \alpha_i \Delta_b$
Reset	yes (at boundaries)	no	no
# Sources	N blocks	$2L$ sublayers	$\sim L/B$ blocks

Table 7 summarizes the differences. **(1) Sources.** Full AttnRes routes over the token embedding and every sublayer output $\mathbf{v}_i$, the same per-sublayer sources as Delta AttnRes. Block AttnRes routes over the embedding, the sum of sublayer outputs within each completed block, and the current intra-block partial sum. Neither uses the cumulative hidden states $\mathbf{h}_i$ measured as a control in Appendix E. Delta Block stores a block-level delta (Appendix C).

```python
def block_attn_res(blocks, partial_block, proj, norm):
    """Replacement routing: weighted sum replaces hidden state."""
    V = torch.stack(blocks + [partial_block])       # [N+1, B, T, D]
    K = norm(V)
    logits = einsum('d, n b t d -> n b t', proj.weight.squeeze(), K)
    h = einsum('n b t, n b t d -> b t d', logits.softmax(0), V)
    return h                                        # replaces hidden
    state

def forward(self, blocks, hidden_states):
    partial_block = hidden_states
    h = block_attn_res(blocks, partial_block, self.attn_res_proj, self
    .attn_res_norm)

    # block boundary: store and reset
    if self.layer_number % (self.block_size // 2) == 0:
        blocks.append(partial_block)
        partial_block = None                        # RESET

    attn_out = self.attn(self.attn_norm(h))
    partial_block = partial_block + attn_out if partial_block is not
    None else attn_out

    h = block_attn_res(blocks, partial_block, self.mlp_res_proj, self.
    mlp_res_norm)
    mlp_out = self.mlp(self.mlp_norm(h))
    partial_block = partial_block + mlp_out
    return blocks, partial_block
```

Figure 7: Original AttnRes (Kimi Team, 2026): replacement routing over the embedding, completed block sums, and the current intra-block partial sum, which restarts at block boundaries.

(2) Routing. AttnRes replaces the hidden state with the weighted sum of sources; delta methods add the weighted sum to the residual stream. **(3) Reset.** AttnRes resets the accumulated within-block residual to zero at each block boundary (at every layer when $N{=}L$), so the next sublayer receives its input only through routing; delta methods never reset.

C Delta Block: Block-Level Variant

The **Delta Block** variant coarsens the per-sublayer sources of Delta AttnRes to block-level granularity. Rather than treating each attention output and each MLP output as a separate source ($2L$ sources for L layers), Delta Block aggregates consecutive sublayer outputs within a contiguous span of B layers into a single block delta

$$\Delta_b = \mathbf{h}_{(b+1)B} - \mathbf{h}_{bB} = \sum_{i \in \text{block } b} \mathbf{v}_i, \tag{3}$$

yielding $B{+}1$ routing sources for a model partitioned into B blocks (plus the current partial block). Equation 3 is the intended block difference; the experiments use the recurrence described below. Routing remains additive: $\hat{\mathbf{h}}_l = \tilde{\mathbf{h}}_l + \sum_b \alpha_{b\to l} \cdot \Delta_b$, with $\alpha_{b\to l} = \mathrm{softmax}(\mathbf{w}_l^\top \mathrm{RMSNorm}(\Delta_b))$.

Source construction used in the experiments. All reported Delta Block experiments use a different recurrence from Equation 3. Let $\mathbf{p}_k$ denote the residual stream at boundary k. The first source is $\mathbf{s}_0 = \mathbf{p}_0$ (the token embedding), and each subsequent source is $\mathbf{s}_k = \mathbf{p}_k - \mathbf{s}_{k-1}$, i.e. it subtracts the preceding *source* rather than the preceding boundary *state*, and expands to the alternating sum

$$\mathbf{s}_k = \mathbf{p}_k - \mathbf{p}_{k-1} + \mathbf{p}_{k-2} - \cdots \pm \mathbf{p}_0. \tag{4}$$

Consecutive sources therefore differ by $\mathbf{p}_k - 2\mathbf{s}_{k-1}$. This recurrence is the one shown in Figure 3. The adjacent-boundary difference $\mathbf{p}_k - \mathbf{p}_{k-1}$ would require retaining the previous boundary state separately; it has not been trained, so whether it improves on the alternating sum remains untested. The source statistics in Appendix E were measured on per-sublayer sources, not on these block sources. Delta AttnRes stores sublayer outputs directly and does not use this recurrence.

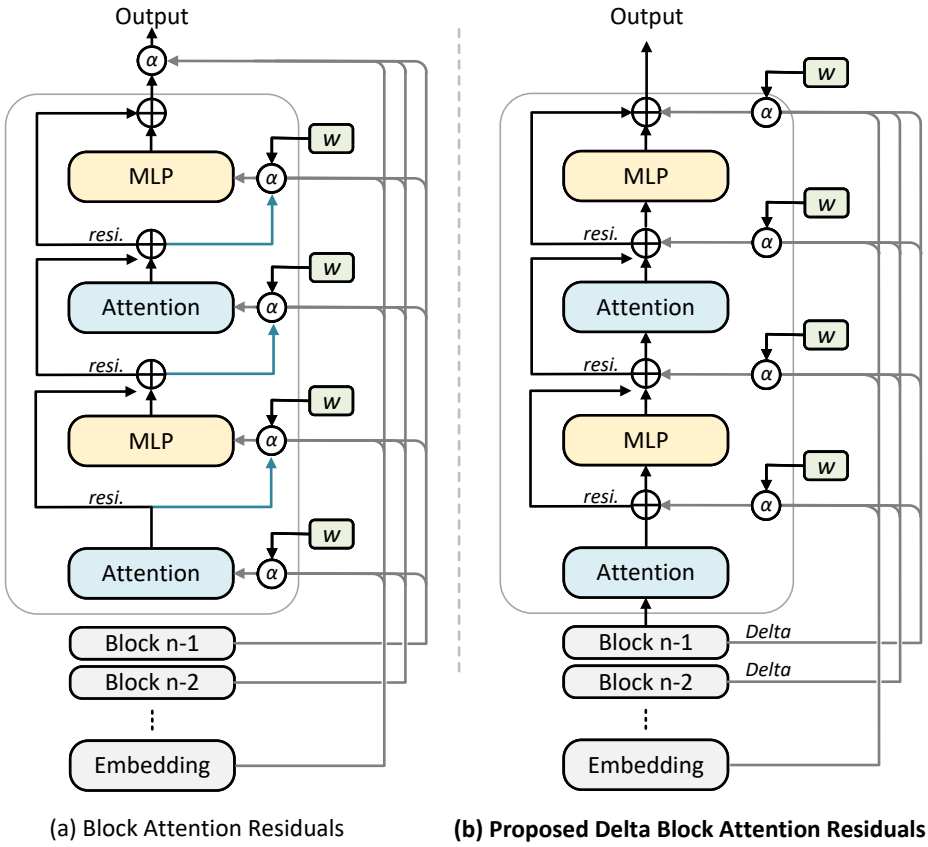

Figure 8: **Delta Block schematic.** Sublayer outputs within each block of B layers are summed into a single block delta Δ_b, which becomes one routing source. The current (in-progress) block contributes a partial delta. The residual stream $\tilde{\mathbf{h}}_l$ is preserved; routing additively augments it.

D Controlled ablation: sources vs. update rule

In the notation of Eq. 1, Full AttnRes is replacement routing over the same per-sublayer delta sources as Delta AttnRes; the two differ only in the update rule. The delta+replacement cell below is a separate control that additionally keeps the running residual stream and offers it as a source. Delta Attention Residuals differ from a cumulative-source router in two ways: the source representation (cumulative $\rightarrow$ delta) and the update rule (replacement $\rightarrow$ additive). To separate them we train the full 2×2, $\{$cumulative, delta$\} \times \{$replacement, additive$\}$, under the recipe of Table 1. All four cells use the same granularity (per-sublayer, $2L$ sources) and match in parameter count to four significant digits, so only the crossed factors differ. We repeat the whole grid on a second architecture family, a LLaMA-style decoder (Appendix F). Repeated runs of the configurations in Table 1 differ from the reported results by at most 0.03 PPL (Delta AttnRes 36.82/31.03 vs. 36.83/31.05; baseline 38.69/32.02 vs. 38.71/32.00). Each cell uses a single seed under the 10K-step protocol. We interpret differences below $\sim$0.3 PPL cautiously; the principal comparisons differ by 1.1–2.1 PPL.

Table 6 in §4.2 reports the grid.

Interaction between source representation and update rule. At 220M, either change alone increases PPL relative to cumulative+replacement ($+0.85$ for delta+replacement, $+1.05$ for cumulative+additive on Qwen). At 533M, each change reduces PPL (-1.51, -1.60), and the combination gives the largest reduction (-2.14). Delta+additive has the lowest PPL in all

four settings. At 533M, the four cells have the same ordering on both architecture families (delta+additive < cumulative+additive < delta+replacement < cumulative+replacement). Comparing cumulative+replacement with delta+additive alone, as in Table 1, does not isolate either factor. The controlled ablation shows that the effect of each choice depends on the other.

E MEASURED SOURCE REDUNDANCY AND ROUTING STATISTICS

We measure source redundancy on the trained checkpoints of Appendix D, using held-out data. At each routing site, we compute the cosine similarity between adjacent sources and the linear predictability (R^2, least squares) of the newest source from all earlier sources.

Table 8: **Source redundancy, measured** at relative depths $0.1/0.5/0.9$ (values given as 220M / 533M). Cumulative sources converge toward exact linear dependence on their own prefix; delta sources stay near-orthogonal and their predictability plateaus around 0.2–0.3.

Metric	Sources	depth 0.1	depth 0.5	depth 0.9
R^2 (newest | earlier)	cumulative	0.46 / 0.45	0.90 / 0.95	0.97 / 0.99
	delta	0.01 / 0.01	0.07 / 0.09	0.22 / 0.28
adjacent-source cosine	cumulative	0.67 / 0.64	0.89 / 0.93	0.93 / 0.97
	delta	-0.03 / 0.06	-0.02 / -0.02	0.01 / 0.00

The same pattern holds on the LLaMA family (adjacent cosine ≈ 0.9 cumulative vs. ≈ 0 delta).

E.1 NORMALIZED ROUTING STATISTICS

Maximum routing weight, shown in Figure 1, depends on source count: under uniform weighting it equals $1/N$. Comparisons across configurations must therefore account for the number of available sources. We report the effective source fraction $N_{\text{eff}}/N = 1/(N\sum_i \alpha_i^2)$, the normalized entropy $H/\ln N$, and the routing-weighted depth-reach $R_l = \sum_i \alpha_{i\to l}(l-i)$ (in source-index units, comparable only at matched N).

Table 9: **Normalized routing statistics** per routing site, reported as average over sites / deep third; discussed in §4.1.

Cell	N_{eff}/N (220M)	N_{eff}/N (533M)	$H/\ln N$ (220M)	depth-reach (220M)
cumulative + replacement	0.47 / 0.33	0.31 / 0.18	0.70 / 0.76	4.6 / 6.6
delta + replacement	0.62 / 0.56	0.52 / 0.45	0.83 / 0.89	5.9 / 10.4
cumulative + additive	0.28 / 0.21	0.18 / 0.17	0.37 / 0.50	7.5 / 11.1
delta + additive	0.25 / 0.19	0.15 / 0.09	0.43 / 0.47	5.1 / 10.7

F LLAMA-FAMILY REPLICATION

The replication uses a standard LLaMA decoder block (pre-norm RMSNorm, RoPE, grouped-query attention, gated-SiLU MLP). The attention and MLP computations retain their standard form. Cross-layer routing uses the same source representations and update rules as the Qwen3 experiments, so the 2×2 cells differ across families only in the backbone. The structurally relevant difference from Qwen3 is the absence of QK-norm: Qwen3 applies per-head RMSNorm to queries and keys, which changes the scale statistics of the residual stream that our account depends on. Configurations mirror the Qwen3 scales (220M: d=768, L=12, 12 heads / 4 KV; 533M: d=1024, L=24, 16 heads / 8 KV; gated-MLP width matched to the Qwen3 intermediate size), and the tokenizer, data stream and shuffling seed, learning rate schedule, step budget and evaluation protocol are held fixed. The rankings, margins, and measured source statistics agree across the two families; other architectures remain untested.

G MEMORY DECOMPOSITION

Per-source tensors have identical shapes across configurations, and Delta Block stores the token embedding as source 0, so peak source counts match at equal block size; the memory differences in Table 1 do not come from source storage. Re-measuring with a decomposition (220M, batch 4, sequence length 1024, forward and backward passes, single GPU, without compilation):

Table 10: **Peak memory decomposition** (GiB). Forward-only peaks are identical across configurations; the difference arises from activations retained for the backward pass.

Configuration	Forward + backward	Forward only	Retained for backward
Baseline	10.78	6.32	4.46
AttnRes (B=4)	13.27	6.32	6.95
Delta Block (B=4)	12.42	6.32	6.09
Delta Block (B=2)	11.77	6.32	5.45

At matched block size (B=4), replacement routing retains $\sim$0.9 GiB more than additive delta routing, because the pre-replacement stream must be kept for the backward pass through the routing operation.

H LIMITATIONS

Scope of evidence. Our systematic comparisons are from-scratch runs at 220M–1B under a 10K-step ($\approx$0.3B-token) protocol; the 8B experiment covers a single configuration (Delta Block), and we have not verified that the orderings persist at convergence-scale token budgets, where the value of cross-layer routing may shrink or grow. Each cell uses a single seed, so differences below $\sim$0.3 PPL should be read with caution. Fine-tuning is evaluated on one base model (Qwen3-0.6B) and one adaptation recipe.

Scope of interpretation. Our routing statistics (N_{eff}, entropy, source similarity) are correlational. Causal evidence is limited to the 2×2 intervention (Appendix D). The proposed explanation that distinguishable sources improve routing remains a hypothesis.

Future work. Further work could examine the interaction between source representation and update rule, and why delta sources remain near-orthogonal during training.