EFFICIENT LANGUAGE MODEL PRETRAINING VIA LOW-RANK READOUT MODULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient pretraining of language models (LMs) requires reaching a desired model quality with fewer optimization steps and minimal additional cost per step. Most work pursues this goal through data curation, improved optimizers, or more efficient Transformer variants. In this work, we focus on a less explored aspect: the LM's readout head. We introduce Low-Rank Readout Modulation (LRM), a simple, lightweight feature-wise gate applied once between the final normalization and tied LM head. LRM derives its scale from the embedding of the observed token, leaves the Transformer stack and KV cache unchanged, and requires neither another decoder pass nor a custom kernel. Across three seeds at 115M, 300M, and 1B scales, LRM improves held-out negative log-likelihood (NLL) on both standard Transformer and Multi-Head Attention Residuals (MHAR) backbones, with mean gains of 0.024–0.043 NLL. It adds at most 0.06% parameters to the base model and, at similar throughput, reaches loss targets with 15–30% fewer steps. Across six backbone–task pairs, parameter-efficient fine-tuning of models pretrained with LRM improves mean accuracy in five.

1 INTRODUCTION

Large language models (LLMs) acquire much of their general-purpose capability during pretraining (Brown et al., 2020), but doing so requires substantial data and compute. This has motivated work on efficient pretraining in several dimensions. Scaling laws and compute-optimal training allocate a fixed budget between parameters and tokens (Kaplan et al., 2020; Hoffmann et al., 2022), while data-centric methods increase the useful signal per token through filtering and curation (Penedo et al., 2024). In parallel, system optimization reduces the cost of each token through IO-aware kernels (Dao et al., 2022). Architectural and optimization methods reduce activated computation with sparse mixture-of-experts models (Fedus et al., 2022), expand residual capacity at low parameter and FLOP cost (Mak & Flanigan, 2025), gate attention outputs (Qiu et al., 2025), or reduce the number of updates through normalized parameterizations (Loshchilov et al., 2025). We pursue a complementary form of architectural efficiency by modulating the output representations to reduce the number of training steps required to reach a target loss while adding negligible cost.

Many efficiency methods change how the Transformer builds its internal representations while leaving the final prediction layer untouched. After the Transformer processes a prefix, its final hidden state is compared with a fixed set of token embeddings to score the possible next tokens. In most LMs, these output embeddings are shared with the input embeddings, known as weight tying (Inan et al., 2017; Press & Wolf, 2017). Although the hidden states change with context, the readout used to interpret them does not. This suggests a complementary route to faster pretraining: make the readout more flexible while leaving the Transformer blocks and autoregressive objective unchanged.

Motivated by this, we introduce Low-Rank Readout Modulation (LRM). At each position, LRM passes the input token embedding to a compact low-rank gate to determine how strongly each feature of the final hidden state should contribute to the next-token prediction. The resulting feature-wise scale is applied after the Transformer's final normalization and immediately before the tied LM head. LRM therefore leaves the attention and feed-forward blocks, the vocabulary projection, and the autoregressive training objective unchanged. We initialize the gate's final projection to zero, making every initial scale neutral and the complete augmented model exactly equivalent to its baseline at the

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.

start of training. The same gate can consequently be added to a standard Transformer or architectural variants such as Multi-Head Attention Residuals (MHAR) (Luo et al., 2026).

Across three seeds at each scale, LRM reduces NLL on both standard Transformer and MHAR backbones by 0.038–0.043 at 115M, approximately 0.024 at 300M, and 0.028–0.031 at 1B scales. Its learning curves reach the corresponding baseline losses 15–30% earlier while adding at most 0.06% parameters and preserving similar training throughput. To understand where the improvement comes from, we compare LRM with equally sized and token-independent alternatives. LRM performs better on average than an equally sized linear head adapter, so parameter count alone does not reproduce its mean gain.

More specifically, our contributions are as follows:

- We introduce LRM, a lightweight feature-wise readout gate that adds $O(Dr)$ computation per token while leaving the Transformer backbone and KV cache unchanged.
- We demonstrate consistent reductions in held-out NLL and earlier target crossing on standard Transformer and MHAR backbones using three seeds at each of 115M, 300M, and 1B scales.
- We perform thorough ablations to examine the effects of multiplicative modulation, low-rank parameterization, and token conditioning.
- We evaluate transfer from the 1B checkpoints, finding lower zero-shot WikiText-2 perplexity on both backbones and higher average LoRA accuracy in five of six comparisons.

2 Method: Low-Rank Readout Modulation

2.1 A token-conditioned readout metric

Let $E \in \mathbb{R}^{V \times D}$ be the tied input/output embedding matrix, where V is the vocabulary size and D is the embedding dimension. At position t, the decoder has observed x_t, and its final normalized state $h_t \in \mathbb{R}^D$ predicts x_{t+1}. A standard Transformer with weight tying maps this state directly to vocabulary logits:

$$\ell_t^{\text{base}} = E h_t \tag{1}$$

LRM modifies only this final readout by applying a token-conditioned feature scale to the final representation before the same tied head:

$$\begin{aligned} a_t &= \begin{cases} 0, & t = 0, \\ \mathrm{RMSNorm}(E[x_t]), & t > 0, \end{cases} \\ b_t &= \mathrm{SiLU}(W_{\text{down}} a_t) \\ g_t &= \tanh(W_{\text{up}} b_t) \\ h'_t &= h_t \odot (1 + g_t), \qquad \ell_t^{\text{LRM}} = E h'_t \end{aligned} \tag{2}$$

where $W_{\text{down}} \in \mathbb{R}^{r \times D}$ and $W_{\text{up}} \in \mathbb{R}^{D \times r}$, with $r \ll D$. We call the mapping from $E[x_t]$ to g_t the LRM gate; it consists of RMSNorm, two low-rank projections, and SiLU and tanh nonlinearities. It is lightweight and does not include the Transformer or tied output head. By convention, we set $a_0 = 0$, so LRM leaves the first position of each packed token sequence unchanged. This affects one position per sequence and is not required for causality. At positions $t \geq 1$, a_t is computed from x_t, which is known when predicting x_{t+1}. The additive 1 in $1 + g_t$ ensures that a zero gate leaves the representation unchanged: if $g_t = 0$, then $h'_t = h_t$. SiLU supplies the bottleneck nonlinearity, and tanh bounds $g_t \in (-1, 1)$, so the applied multiplier $1 + g_t$ lies in $(0, 2)$. RMSNorm removes source magnitude (Zhang & Sennrich, 2019). The forwad pass of LRM is shown in Algorithm 1.

2.2 Identity-preserving initialization

Following the asymmetric initialization used for LoRA's low-rank factors (Hu et al., 2022), we initialize W_{down} randomly and set the final projection $W_{\text{up}} = 0$. Consequently, the gate initially outputs zero, so the logits and loss exactly match the baseline. At initialization, only W_{up} receives a nonzero loss gradient; the rest of the gate receives loss gradients after W_{up} becomes nonzero.

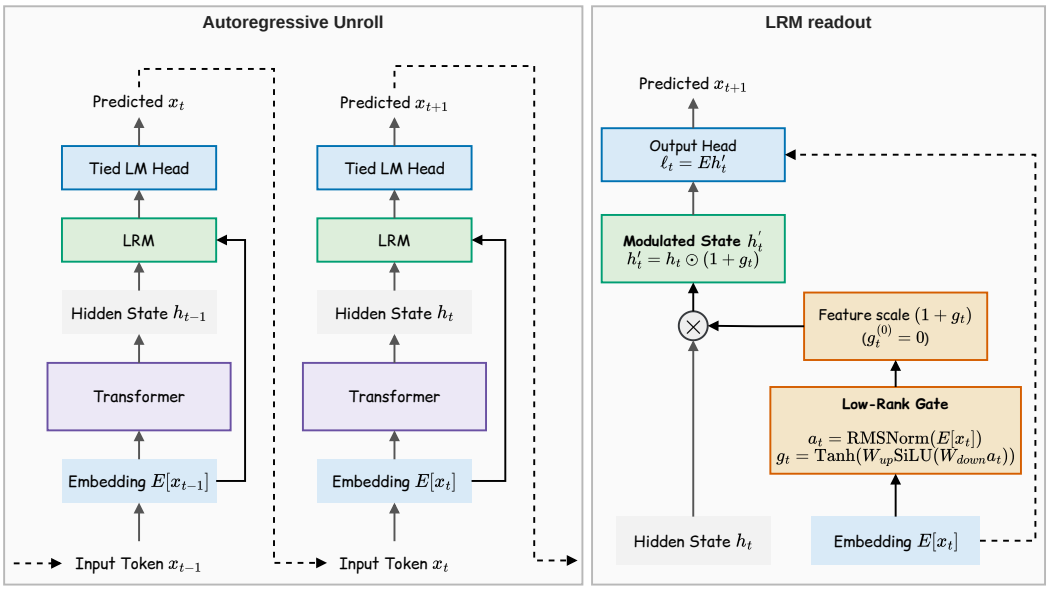

Figure 1: **Overview of LRM.** *Left:* The observed token x_t follows the standard autoregressive path, while its embedding also conditions LRM. *Right:* The low-rank gate produces $1 + g_t$, which rescales h_t before the tied head predicts x_{t+1}. Dashed lines denote token flow and weight tying. Zero initialization exactly recovers the baseline; the Transformer and KV cache are unchanged.

Algorithm 1 Token-conditioned LRM coupled to a Transformer

Require: Tokens x; tied embedding E; Transformer F; LRM weights $W_{\mathrm{down}}, W_{\mathrm{up}}$, with $W_{\mathrm{up}} = 0$ initially

$Z \leftarrow E[x]$ {input-token embeddings}

$H \leftarrow F(Z)$ {final normalized Transformer states}

$A \leftarrow \mathrm{RMSNorm}(Z)$; set $A_{:,0,:} \leftarrow 0$

$G \leftarrow \tanh\big(\mathrm{SiLU}(AW_{\mathrm{down}}^{\top})W_{\mathrm{up}}^{\top}\big)$

return $[H \odot (1 + G)]\, E^{\top}$ {next-token logits}

This is also related to identity-initialized residual parameterizations such as ReZero (Bachlechner et al., 2021), LayerScale (Touvron et al., 2021), and adaLN-Zero (Peebles & Xie, 2023). LRM adds $2Dr + D$ parameters and approximately $4Dr$ projection FLOPs per token when multiplication and addition are counted separately. It does not require any modifications to attention or feed-forward layers and requires no custom kernel. At inference time, the token-dependent bottleneck can be precomputed in a vocabulary-by-rank matrix. For details on latency and memory see Appendix D.

2.3 Composition with MHAR

We couple LRM with a standard Transformer decoder as shown in Figure 1. We also couple it with MHAR (Luo et al., 2026) to study its effect on other variants. An ordinary Transformer carries one residual stream forward by adding each attention and MLP update. MHAR instead maintains the input embedding and the outputs of preceding attention and MLP layers as separate representations. Before each subsequent layer, it forms the input as a learned weighted combination of these representations. For available sources $S = (v_1, \ldots, v_m)$, routing head k forms:

$$R_k(S) = \sum_{i=1}^{m} \mathrm{softmax}_i\big(q_k^{\top}[\mathrm{RMSNorm}(v_i)]_k\big)\, v_{i,k} \tag{3}$$

where the softmax is taken across depth sources i. The learned query $q_k \in \mathbb{R}^{D/H}$ scores the normalized slice assigned to head k, and concatenating the H independently routed slices restores width D. LRM does not alter this routing. Its input h_t is the normalized output of MHAR's terminal router, and Equation 2 applies the modulation immediately before the same tied output head.

3 Analysis: What LRM Computes

Although LRM leaves the vocabulary projection unchanged, its multiplicative form changes the geometry of the readout. Let e_j denote row j of E, and define $M_t = \mathrm{diag}(1 + g_t)$. The logit for candidate token j can then be written as:

$$\ell_{t,j} = e_j^\top M_t h_t = [e_j \odot (1 + g_t)]^\top h_t \tag{4}$$

Because every diagonal entry of M_t lies in $(0, 2)$, Equation 4 equips the readout with a positive diagonal metric selected by the observed token. It can amplify or attenuate a feature's contribution, but cannot reverse its sign. Because M_t is invertible, it also preserves the head's column space and nullspace:

$$\mathrm{col}(EM_t) = \mathrm{col}(E), \qquad \ker(M_t E^\top) = \ker(E^\top) \tag{5}$$

This shows that LRM reweights the D directions already transmitted by the tied head. It neither increases the head's rank nor restores directions removed by the vocabulary projection. This distinguishes LRM from rank-expanding output layers. Recent work separately studies whether the LM head restricts backward gradient flow (Godey & Artzi, 2026; Murugan, 2026). Our analysis concerns the geometry of the forward readout.

The geometry above characterizes LRM's representational effect. To understand how the gate begins learning, consider the early regime in which $z_t = W_{\mathrm{up}}\phi(W_{\mathrm{down}}a_t)$ is small:

$$\Delta\ell_{t,j} \approx (e_j \odot h_t)^\top W_{\mathrm{up}}\phi(W_{\mathrm{down}}a_t) \tag{6}$$

where $\phi = \mathrm{SiLU}$. In this approximation, the correction depends jointly on the candidate-token embedding e_j, the final state h_t, and the observed-token representation a_t. This is related to broader low-interaction-rank analyses of multiplicative heads (Zhao & Li, 2026). It also shows that an equally sized additive modulator contributes $e_j^\top g_t$, whereas LRM contributes $(e_j \odot h_t)^\top g_t$.

4 Related Work

Tied embeddings and output parameterization. Weight tying uses the same matrix for input embeddings and output classifier weights (Inan et al., 2017; Press & Wolf, 2017). This reduces parameters and aligns the two token spaces, but forces each token vector to serve both contextual representation and next-token discrimination, even when those objectives favor different updates (Lopardo et al., 2026). Embedding decoupling (Chung et al., 2021), joint input–output spaces (Pappas et al., 2018), Leviathan (Batley & Saha, 2026), and pseudo-inverse tying (Gu et al., 2026) alter that coupling, whereas RiLM removes the linear head entirely (Li, 2026). Rank-expanding output layers instead alter the softmax parameterization (Yang et al., 2018; Takase et al., 2018; Pappas & Henderson, 2019), while ARCHead compresses a pretrained head (Kocabay et al., 2026). LRM retains one tied matrix and one softmax, making only the pre-head feature metric context dependent.

Conditional gating and token reinjection. The closest predecessor to LRM is Input-to-Output Gate (IOG), which uses the current token to gate RNN logits (Takase et al., 2017). With vocabulary gate v_t, IOG applies $\mathrm{diag}(v_t)Eh_t$, whereas LRM applies $E\,\mathrm{diag}(1 + g_t)h_t$. IOG learns its vocabulary-space gate from a separate token embedding matrix and trains it post hoc with the RNN frozen, whereas LRM jointly trains a rank-r feature-space gate from the tied embedding. Conditional BatchNorm (de Vries et al., 2017), conditional instance normalization (Dumoulin et al., 2017), AdaIN (Huang & Belongie, 2017), FiLM (Perez et al., 2018), and adaLN-Zero (Peebles & Xie, 2023) establish feature-wise affine conditioning and zero-initialized identity branches. LRM instead relocates a scale-only version to the LM readout. JTok is the closest Transformer analogue: it uses a normalized, token-indexed identity-offset gate at every MLP residual (Yang et al., 2026). LRM instead derives a rank-r code from the existing tied embedding and acts once after the final normalization. IA3 (Liu et al., 2022) learns input-independent vectors that rescale attention and feed-forward activations during parameter-efficient adaptation. LRM, on the other hand, is token-conditioned and scales only the final normalized hidden state before the LM head.

Depth routing, recurrence, and auxiliary prediction. Attention Residuals (Kimi Team et al., 2026) and MHAR (Luo et al., 2026) learn to mix representations across depth. MHAR routes depth

sources before the final normalization whereas LRM scales features afterward. Residual Matrix Transformers (RMTs) replace each vector residual stream with an outer-product memory and use learned keys to read and write it throughout the stack, including at the final unembedding (Mak & Flanigan, 2025). RMT is a backbone-wide architecture with separate input and output embeddings, whereas LRM leaves the vector residual stream and all Transformer blocks unchanged and modulates the state once before a tied head. Recurrent and feedback Transformers carry high-level states into later stack computations (Fan et al., 2021; Huang et al., 2026; Cai et al., 2026; Wang et al., 2026) whereas LRM preserves one parallel stack pass.

5 Experimental Setup

We train Qwen3-style decoder-only models (Yang et al., 2025) from scratch on FineWeb-Edu (Penedo et al., 2024) at 115M, 300M, and 1B parameter scales. The 115M model has width 512, 12 layers, 8 query heads, 4 key–value heads, and feed-forward width 1,536. The 300M model has width 768, 27 layers, 12 query heads, 4 key–value heads, and feed-forward width 2,304. The 1B model has width 1,536, 28 layers, 24 query heads, 8 key–value heads, and feed-forward width 4,608. All models use the `Qwen/Qwen3-0.6B` tokenizer with a vocabulary of 151,936. They use tied input–output embeddings and sequence length of 2,048. All models are trained with the same next-token prediction objective. At every scale, we evaluate a standard Transformer and MHAR, each without or with LRM. MHAR uses eight routed slices at each attention input, MLP input, and terminal read.

Every run uses 10,000 optimizer steps and an effective global batch of 64 sequences, totaling 1.31B sequence tokens. Training uses BF16 and AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at 1.0. Within each scale, tokenizer, data order, optimizer, schedule, batch construction, validation stream, and evaluation cadence are identical across all four configurations. The 115M and 300M runs use peak learning rate 10^{-3}, 100 warmup steps, and cosine decay to 10^{-4} over the full horizon. The 1B runs use peak learning rate 5×10^{-4}, 100 warmup steps, and decay through step 2,500 to a 5×10^{-5} floor. Ranks are 64 at 115M and 300M and 128 at 1B. Following LoRA-style initialization (Hu et al., 2022), we draw $W_{\text{down}} \sim \mathcal{N}(0, 0.02^2)$, set $W_{\text{up}} = 0$, initialize the conditioning RMSNorm scale to one, use $\epsilon = 10^{-6}$, and omit projection biases.

We evaluate the model every 250 steps at 115M and every 500 steps at 300M and 1B. For NLL, we report the mean of the final three evaluations. We measure how many training steps LRM needs to match the loss reached after 10,000 steps by the same model trained without LRM. To reduce evaluation noise, we compare three-evaluation moving averages.

6 Results

6.1 Pretraining quality and efficiency

Table 1 reports the complete four-model comparison. At 115M, LRM improves the standard Transformer by 0.043 ± 0.004 NLL and MHAR by 0.038 ± 0.006 (mean $\pm$ SD across three seeds). The three-seed mean curves for Transformer + LRM and MHAR + LRM both reach their respective baseline targets at step 7,250, requiring 27.5% fewer updates.

After accounting for measured throughput, the estimated training-time reductions are 25.5% and 27.5%, respectively. The combined MHAR + LRM reduction is 0.102 NLL relative to the Transformer baseline: MHAR accounts for 0.064 (62.7%), while adding LRM to MHAR accounts for a further 0.038 (37.3%). Although smaller in absolute NLL, LRM's incremental gain requires only 0.057% additional parameters and has a measured MHAR+LRM/MHAR throughput ratio of 1.0. It therefore provides more than one-third of the combined improvement at negligible marginal training cost. Table 2 shows that LRM adds at most 0.057% parameters and stays within 3.1% of baseline throughput. Figure 2 compares the corresponding held-out learning trajectories.

At 300M, LRM improves the Transformer backbone by 0.024 ± 0.002 NLL and MHAR by 0.024 ± 0.010. The three-seed mean curves reach both targets after 15% fewer updates. At 1B, LRM improves the Transformer backbone by 0.031 ± 0.006 and MHAR by 0.028 ± 0.012. Both three-seed mean curves reach their baseline terminal targets at step 7,000, 30% before the fixed 10,000-step horizon.

Table 1: Pretraining results. NLL is the final-three-evaluation mean over three seeds.

Scale	Method	Final-window NLL ↓	Δ vs. Transformer ↓	LRM effect ↓
115M	Transformer	3.469 ± 0.004	—	—
	Transformer + LRM	3.426 ± 0.003	-0.043 ± 0.004	-0.043 ± 0.004
	MHAR	3.405 ± 0.004	-0.064 ± 0.007	—
	MHAR + LRM	3.367 ± 0.003	-0.102 ± 0.001	-0.038 ± 0.006
300M	Transformer	3.233 ± 0.004	—	—
	Transformer + LRM	3.210 ± 0.002	-0.024 ± 0.002	-0.024 ± 0.002
	MHAR	3.156 ± 0.006	-0.077 ± 0.003	—
	MHAR + LRM	3.132 ± 0.004	-0.102 ± 0.008	-0.024 ± 0.010
1B	Transformer	3.512 ± 0.017	—	—
	Transformer + LRM	3.481 ± 0.015	-0.031 ± 0.006	-0.031 ± 0.006
	MHAR	3.336 ± 0.001	-0.176 ± 0.017	—
	MHAR + LRM	3.308 ± 0.011	-0.204 ± 0.005	-0.028 ± 0.012

Table 2: Parameter and training efficiency of LRM. For each backbone and scale, the target is the baseline's mean NLL over its final three evaluations. We report when the corresponding LRM mean curve first reaches that value. Time saved accounts for both target-crossing step and measured throughput.

Scale	Baseline	Rank	Added params	Params overhead	Train throughput	Time saved
115M	Transformer	64	66,048	0.057%	0.97	25.5%
	MHAR	64	66,048	0.057%	1.00	27.5%
300M	Transformer	64	99,072	0.033%	0.97	12.3%
	MHAR	64	99,072	0.033%	1.00	15.3%
1B	Transformer	128	394,752	0.039%	0.98	28.5%
	MHAR	128	394,752	0.039%	1.00	30.1%

6.2 Ablation Study

We ablate four design choices at 115M scale.

1. The additive variant replaces feature-wise multiplication, $h'_t = h_t \odot (1 + g_t)$, with addition, $h'_t = h_t + g_t$, testing whether rescaling the final representation is important.
2. The parameter-matched linear head adapter inserts a zero-initialized diagonal-plus-low-rank linear map before the tied output head, testing whether an equal number of output-side parameters is sufficient.
3. The direct static scale learns one token-independent D-dimensional scale vector. The factorized static gate instead feeds the same fixed all-ones vector through the LRM gate at every position. These two controls isolate the effect of low-rank factorization from token conditioning.
4. The detached variant has exactly the same token-conditioned forward computation as LRM, but stops gradients through the token embedding on the gate branch. The embedding remains trainable through its ordinary input and output paths.

Table 3 compares LRM with the direct static scale, factorized static gate, and parameter-matched linear head adapter after 10,000 training steps.

Across the three seeds, multiplicative LRM beats the equal-parameter additive gate by 0.021–0.027 NLL. The 66,048-parameter linear head adapter improves NLL on the Transformer backbone by 0.032 ± 0.009, while LRM improves it by 0.043 ± 0.004. The factorized static gate and detached token-conditioned gate nevertheless recover much of the improvement. These successful variants all use identity-preserving, factorized multiplicative terminal modulation. On the Transformer backbone, conditioning that modulation on the observed token provides a further 0.006 ± 0.005 improvement over the factorized static gate.

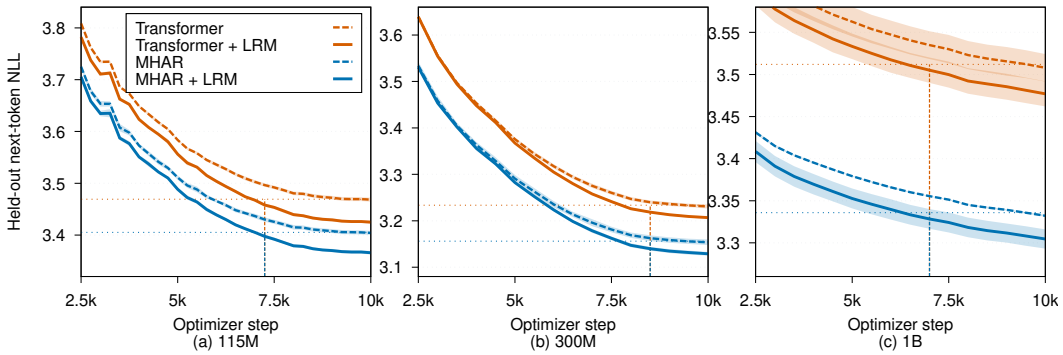

Figure 2: Held-out NLL for Transformer, Transformer + LRM, MHAR, and MHAR + LRM at 115M, 300M, and 1B after step 2,500. Curves show the mean over three seeds. The Transformer and MHAR crossings occur at steps 7,250/7,250, 8,500/8,500, and 7,000/7,000.

Table 3: Three-seed (mean $\pm$ SD) Δ NLL at 115M (more negative is better). The linear adapter was evaluated on the Transformer backbone only.

Readout modification	Added params	Transformer ΔNLL$\downarrow$	MHAR ΔNLL$\downarrow$
Direct zero-initialized scale	512	-0.014 ± 0.002	-0.012 ± 0.006
Factorized static scale	66,048	-0.037 ± 0.005	-0.035 ± 0.008
Token-conditioned LRM	66,048	-0.043 ± 0.004	-0.038 ± 0.006
Linear head adapter	66,048	-0.032 ± 0.009	—

What the static control reveals. The factorized static gate receives the same fixed input at every noninitial position, so after training it applies one learned feature-scale vector, $1 + u$, to each corresponding final hidden state. This scale can be absorbed exactly into the learned scale γ_f of the final RMSNorm at every gated position. For every position $t \geq 1$:

$$\mathrm{RMSNorm}(y_t; \gamma_f) \odot (1 + u) = \mathrm{RMSNorm}(y_t; \gamma_f \odot (1 + u)) \tag{7}$$

Thus, at gated positions, the static gate is functionally equivalent to rescaling the final RMSNorm weights. It changes the parameterization used to learn this scale without introducing token-dependent behavior. Despite this equivalence, the factorized static gate beats a directly parameterized, zero-initialized 512-dimensional scale by 0.023 ± 0.005 NLL on the Transformer backbone and 0.023 ± 0.012 on MHAR. The low-rank factorization therefore improves optimization, consistent with reparameterization and overparameterization (Salimans & Kingma, 2016; Arora et al., 2018).

Full LRM keeps this factorized path but replaces the fixed input with the observed-token embedding, allowing the feature scale to vary across tokens and positions. This token-conditioned gate improves over static on the Transformer backbone by 0.006 ± 0.005. For details, see Appendix C.

6.3 Does LRM transfer to adaptation and a held-out corpus?

We freeze the 1B Transformer and MHAR pairs and train identically configured LoRA adapters (Hu et al., 2022) on PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), and WinoGrande (Sakaguchi et al., 2020). We also evaluate the frozen checkpoints in a zero-shot setting on WikiText-2 (Merity et al., 2017). Table 4 summarizes both evaluations.

LRM improves all three mean accuracies on the Transformer backbone and two of three on MHAR. The clearest gains are on PIQA (+1.45/+1.70 points on Transformer/MHAR). It also reduces WikiText-2 perplexity by 3.32/4.05. Because adapter seeds share one pretraining seed per pretrained configuration, these results establish adapter-initialization robustness, not pretraining-seed uncertainty. For details, see Appendix B.2.

Table 4: Transfer from 1B checkpoints. Classification is LoRA accuracy over three adapter seeds (mean ± SD); WikiText-2 is zero-shot word perplexity.

Pretraining backbone	LoRA accuracy (%) ↑			Zero-shot PPL ↓
	PIQA	BoolQ	WinoGrande	WikiText-2
Transformer	60.75 ± 0.03	66.91 ± 0.29	50.80 ± 0.20	82.49
Transformer+LRM	62.21 ± 0.32	67.61 ± 0.41	51.57 ± 1.07	79.17
Δ_{LRM}	$+1.45 \pm 0.31$	$+0.70 \pm 0.70$	$+0.76 \pm 1.26$	-3.32
MHAR	60.79 ± 0.36	65.65 ± 0.25	52.20 ± 0.99	65.30
MHAR+LRM	62.50 ± 0.44	66.33 ± 0.37	51.22 ± 1.26	61.26
Δ_{LRM}	$+1.70 \pm 0.08$	$+0.68 \pm 0.34$	-0.97 ± 2.20	-4.05

7 Discussion & Limitations

Across model sizes and backbones, LRM consistently improves final-window NLL and reaches the corresponding baseline loss earlier. Because it is applied only once, after the final normalization, these gains require neither changes to the Transformer blocks nor another pass. The similar training throughput and small parameter overhead therefore make LRM a lightweight and practical technique to improve the observed quality–compute tradeoff. Its gains on top of MHAR further show that readout modulation complements, rather than replaces, improved residual-stream construction.

The ablations show that multiplicative modulation outperforms matched addition and has a larger mean gain than an equal-parameter linear head adapter, while the factorized static and detached variants retain most of the gain. Together, these results identify the identity-preserving, factorized terminal scale as an important source of the improvement. Conditioning that scale on the observed token provides an additional gain on the Transformer backbone and remains a useful component of LRM.

Our experiments cover one Qwen-style decoder family, evaluated with either standard residual connections or MHAR, one data recipe, and 1.31B tokens per run (11.3, 4.3, and 1.3 tokens per parameter at 115M, 300M, and 1B). They therefore establish a fixed-budget optimization result, not a compute-optimal scaling law (Kaplan et al., 2020; Hoffmann et al., 2022). As future work, we plan to evaluate LRM at larger scales and compute-optimal token budgets, characterize when token conditioning improves over static modulation, and extend the post-training study beyond LoRA adaptation to instruction tuning and reinforcement learning.

8 Conclusion

We introduced Low-Rank Readout Modulation, a low-rank gate that improves pretraining efficiency by rescaling the final normalized representation before the tied LM head. LRM leaves attention, feed-forward layers, and the KV cache unchanged. Across three seeds at 115M, 300M, and 1B parameter scales, every Transformer and MHAR comparison favors LRM: mean NLL improves by 0.024–0.043, while the corresponding baseline loss is reached with 15–30% fewer updates and at most 0.06% additional parameters. The benefit also persists in zero-shot WikiText-2 evaluation and in five of six mean LoRA fine-tuning experiments.

The ablations strengthen this conclusion. Multiplicative LRM outperforms matched addition and has a larger mean gain than an equal-parameter linear adapter, showing that equal parameter count alone does not reproduce its mean improvement. A factorized static gate delivers substantial gains, showing that factorized terminal modulation accounts for much of the improvement; conditioning it on the observed token further improves the Transformer results. Taken together, the results establish final hidden state modulation as a simple and inexpensive design principle for improving the quality–compute tradeoff of LM pretraining.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, et al. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2204.14198.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1802.06509.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2021. URL https://arxiv.org/abs/2003.04887.

Reza T. Batley and Sourav Saha. Leviathan: Decoupling input and output representations in language models. *arXiv preprint arXiv:2601.22040*, 2026. URL https://arxiv.org/abs/2601.22040.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023. URL https://arxiv.org/abs/2303.08112.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1911.11641.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2005.14165.

Ziyang Cai, Xingyu Zhu, Yihe Dong, Yinghui He, and Sanjeev Arora. T^2mlr: Transformer with temporal middle-layer recurrence. *arXiv preprint arXiv:2607.15178*, 2026. URL https://arxiv.org/abs/2607.15178.

Hyung Won Chung, Thibault Févry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. Rethinking embedding coupling in pre-trained language models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.12821.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019. URL https://arxiv.org/abs/1905.10044.

Jan Christian Blaise Cruz and Alham Fikri Aji. Sense representations are inducible interfaces. *arXiv preprint arXiv:2605.28669*, 2026. URL https://arxiv.org/abs/2605.28669.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.14135.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of ICML*, 2017. URL https://arxiv.org/abs/1612.08083.

Soham De and Samuel L. Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2002.10444.

Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1707.00683.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1610.07629.

Denis Emelin, Ivan Titov, and Rico Sennrich. Widening the representation bottleneck in neural machine translation with lexical shortcuts. In *Proceedings of the Fourth Conference on Machine Translation*, 2019. URL https://arxiv.org/abs/1906.12284.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2002.09402.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://arxiv.org/abs/2101.03961.

Nathan Godey and Yoav Artzi. Lost in backpropagation: The LM head is a gradient bottleneck. *arXiv preprint arXiv:2603.10145*, 2026. URL https://arxiv.org/abs/2603.10145.

Jian Gu, Aldeida Aleti, Chunyang Chen, and Hongyu Zhang. Rethinking weight tying: Pseudo-inverse tying for lm stable training and updates. *arXiv preprint arXiv:2602.04556*, 2026. URL https://arxiv.org/abs/2602.04556.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.09106.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on LoRA finetuning dynamics. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.08447.

John Hewitt, John Thickstun, Christopher D. Manning, and Percy Liang. Backpack language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://arxiv.org/abs/2305.16765.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2203.15556.

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. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. URL https://arxiv.org/abs/1703.06868.

Zeyi Huang, Xuehai He, Liliang Ren, Yiping Wang, Baolin Peng, Hao Cheng, Shuohang Wang, Pengcheng He, Jianfeng Gao, Yong Jae Lee, and Yelong Shen. Latent recurrent transformer: Architecture exploration, training strategies, and scaling behavior. *arXiv preprint arXiv:2605.26797*, 2026. URL https://arxiv.org/abs/2605.26797.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01462.

Jaber Jaber and Osama Jaber. Ouroboros: Dynamic weight generation for recursive transformers via input-conditioned LoRA modulation. *arXiv preprint arXiv:2604.02051*, 2026. URL https://arxiv.org/abs/2604.02051.

Ajay Jaiswal, Lauren Hannah, Han-Byul Kim, Duc Hoang, Mehrdad Farajtabar, and Minsik Cho. TIDE: Every layer knows the token beneath the context. *arXiv preprint arXiv:2605.06216*, 2026. URL https://arxiv.org/abs/2605.06216.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Kimi Team et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026. URL https://arxiv.org/abs/2603.15031.

Şuayp Talha Kocabay, Talha Rüzgar Akkuş, and Kamer Ali Yuksel. ARCHead: Activation-metric residual correction for large language model output heads. *arXiv preprint arXiv:2608.02703*, 2026. URL https://arxiv.org/abs/2608.02703.

Fang Li. RiLM: Parameter-efficient language modeling via geodesic decoding. *arXiv preprint arXiv:2609.10305*, 2026. URL https://arxiv.org/abs/2609.10305.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.05638.

Antonio Lopardo, Avyukth Harish, Catherine Arnett, and Akshat Gupta. Weight tying biases token embeddings towards the output space. *arXiv preprint arXiv:2603.26663*, 2026. URL https://arxiv.org/abs/2603.26663.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.01131.

Cheng Luo, Zefan Cai, and Junjie Hu. Multi-head attention residuals. *arXiv preprint arXiv:2607.27230*, 2026. URL https://arxiv.org/abs/2607.27230.

Brian Mak and Jeffrey Flanigan. Residual matrix transformers: Scaling the size of the residual stream. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://arxiv.org/abs/2506.22696.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Anand Murugan. Does the lm head create a harmful gradient bottleneck? a causal test. *arXiv preprint arXiv:2608.16671*, 2026. URL https://arxiv.org/abs/2608.16671.

Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. DenseFormer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.02622.

Nikolaos Pappas and James Henderson. Deep residual output layers for neural language generation. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. URL https://arxiv.org/abs/1905.05513.

Nikolaos Pappas, Lesly Miculicich Werlen, and James Henderson. Beyond weight tying: Learning joint input-output embeddings for neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 73–83, 2018. URL https://arxiv.org/abs/1808.10681.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2212.09748.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024. URL https://arxiv.org/abs/2406.17557.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of AAAI*, 2018. URL https://arxiv.org/abs/1709.07871.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017. URL https://arxiv.org/abs/1608.05859.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Nonlinearity, sparsity, and attention-sink-free. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.06708.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1907.10641.

Tim Salimans and Durk P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://arxiv.org/abs/1602.07868.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. URL https://arxiv.org/abs/1505.00387.

Jonathan Su. Attention projection mixing with exogenous anchors. *arXiv preprint arXiv:2601.08131*, 2026. URL https://arxiv.org/abs/2601.08131.

Sho Takase, Jun Suzuki, and Masaaki Nagata. Input-to-output gate to improve rnn language models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pp. 43–48, 2017. URL https://aclanthology.org/I17-2008/.

Sho Takase, Jun Suzuki, and Masaaki Nagata. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://arxiv.org/abs/1808.10143.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL https://arxiv.org/abs/2103.17239.

Xi Wang, Ziyang Cai, Zheng Zhan, Harry Dong, Ying Fan, Gustavo de Rosa, Tim Pearce, and John Langford. Full-bandwidth transformer. *arXiv preprint arXiv:2608.08888*, 2026. URL https://arxiv.org/abs/2608.08888.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1606.06630.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Yebin Yang, Huaijin Wu, Fu Guo, Lin Yao, Xiaohan Qin, Jingzhi Wang, Debing Zhang, and Junchi Yan. JTok: On token embedding as another axis of scaling law via joint token self-modulation. *arXiv preprint arXiv:2602.00800*, 2026. URL https://arxiv.org/abs/2602.00800.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1711.03953.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1910.07467.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1901.09321.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. LLaMA-Adapter: Efficient fine-tuning of language models with zero-init attention. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2303.16199.

Zijian Zhao and Sen Li. Low-interaction-rank learning: Unifying multiplicative dual-encoder heads. *arXiv preprint arXiv:2608.11661*, 2026. URL https://arxiv.org/abs/2608.11661.

A Technical and Positioning Details

A.1 Extended related-work positioning

Table 5 summarizes the differences among the closest modulation mechanisms.

Table 5: Positioning against the closest modulation mechanisms. "Joint" means trained with the backbone from scratch in the cited setting.

Method	Signal	Modulated site	Exact identity init.	Training
IOG	token table	vocabulary logits	unspecified	frozen RNN
JTok	token table	each MLP residual	unspecified	joint LM
adaLN-Zero	condition	block normalization	yes	joint diffusion
IA3	static scale	attention/FFN	n/a	adaptation
LRM (ours)	tied token embedding	post-norm readout	yes	joint LM

Conditional BatchNorm (de Vries et al., 2017), conditional instance normalization (Dumoulin et al., 2017), AdaIN (Huang & Belongie, 2017), FiLM (Perez et al., 2018), and adaLN-Zero (Peebles & Xie, 2023) establish feature-wise affine conditioning. ReZero (Bachlechner et al., 2021), LayerScale (Touvron et al., 2021), Fixup (Zhang et al., 2019), SkipInit (De & Smith, 2020), Flamingo (Alayrac et al., 2022), and LLaMA-Adapter (Zhang et al., 2024) show that zero- or small-initialized branches can change optimization while preserving or closely approximating the initial function. LRM adopts this optimization pattern and places a scale-only branch at the post-normalization LM readout.

Highway networks (Srivastava et al., 2015), gated linear units (Dauphin et al., 2017), multiplicative integration (Wu et al., 2016), and HyperNetworks (Ha et al., 2017) are broader feature-modulation antecedents. Lexical Shortcuts (Emelin et al., 2019), DenseFormer (Pagliardini et al., 2024), ExoFormer (Su, 2026), and TIDE (Jaiswal et al., 2026) reuse token or earlier-layer representations inside the stack; Ouroboros conditionally modulates LoRA bases across recursive depth (Jaber & Jaber, 2026). LRM uses the aligned token once at the readout, leaving attention and cached states unchanged.

Backpack (Hewitt et al., 2023) and ACROS (Cruz & Aji, 2026) construct predictions or terminal residuals from embedding-derived representations; neither implements LRM's post-normalization low-rank diagonal metric. Tuned Lens learns diagnostic intermediate decoders and leaves the model's predictive path unchanged (Belrose et al., 2023). LRM's contribution is the resulting composite design rather than any individual gating ingredient.

B Additional Results

B.1 Conditioning-source comparison

We compared four causal signals: (1) the observed-token embedding, (2) the previous final state, (3) the observed-token embedding minus the previous final state, and (4) the current final state. All used the same low-rank readout. Their close ordering in Table 6 motivated the subsequent ablations that isolate the contribution of token conditioning.

Table 6: Conditioning-source comparison at 500 updates. ΔNLL is relative to Transformer; negative is better.

Conditioning signal	Final NLL $\downarrow$	ΔNLL $\downarrow$
Transformer	5.065	—
Observed-token embedding	4.909	-0.156
Previous state	4.911	-0.154
Current final state	4.917	-0.148
Token embedding minus previous state	4.923	-0.142
Observed-token embedding, detached	4.975	-0.090

At the full 10,000-step horizon on the 115M MHAR backbone, detached LRM improves NLL over MHAR by 0.042 ± 0.003 across three seeds, compared with 0.038 ± 0.006 for attached LRM. Thus, the forward modulation retains its benefit without a gradient path from the gate into the conditioning embedding.

B.2 Post-training composition

The transfer comparison starts from the rank-128, 10,000-step, 1B Transformer and MHAR pairs in Table 1. All four pretrained models are frozen, including LRM. PIQA, BoolQ, and WinoGrande use all-layer Q/V LoRA rank 32 at 10^{-3}, selected in an earlier 300M hyperparameter sweep and then held fixed for the 1B comparison.

Table 7: Response NLL after LoRA adaptation of the frozen 1B checkpoints (mean $\pm$ SD).

Pretraining backbone	PIQA↓	BoolQ↓	WinoGrande↓
Transformer + LoRA	3.431 ± 0.002	0.305 ± 0.000	0.763 ± 0.004
Transformer + LRM + LoRA	3.391 ± 0.002	0.302 ± 0.001	0.697 ± 0.002
Δ_{LRM}	-0.040 ± 0.001	-0.004 ± 0.001	-0.066 ± 0.002
MHAR + LoRA	3.284 ± 0.001	0.309 ± 0.001	0.594 ± 0.002
MHAR + LRM + LoRA	3.218 ± 0.003	0.308 ± 0.001	0.590 ± 0.010
Δ_{LRM}	-0.066 ± 0.002	-0.001 ± 0.001	-0.004 ± 0.012

LRM-pretrained checkpoints reduce mean response NLL in all six comparisons. The largest reductions appear on PIQA for both backbones and on WinoGrande for the Transformer backbone.

C Static Modulation and Rank Controls

For the token-invariant gate, let γ_s be the conditioning RMSNorm weight. The fixed all-ones input produces a static bottleneck code and gate:

$$b_{\mathrm{static}} = \mathrm{SiLU}\left(W_{\mathrm{down}} \frac{\gamma_s}{\sqrt{1+\epsilon}}\right), \qquad u = \tanh(W_{\mathrm{up}} b_{\mathrm{static}}) \tag{8}$$

If the final RMSNorm has weight γ_f, then at every non-initial position we have:

$$\mathrm{RMSNorm}(y_t; \gamma_f) \odot (1+u) = \mathrm{RMSNorm}(y_t; \gamma_f \odot (1+u)) \tag{9}$$

Because u is constant, its effect can be absorbed into the learned final RMSNorm scale γ_f at every gated position. The static gate therefore changes how the final feature scale is optimized without adding a new token-dependent mapping at those positions. It provides an additional zero-initialized optimization path, consistent with prior results on reparameterization and overparameterization (Salimans & Kingma, 2016; Arora et al., 2018; Hayou et al., 2024). At initialization, $W_{\mathrm{up}} = 0$, so only the up projection receives a nonzero loss gradient. Once W_{up} becomes nonzero, gradients also reach W_{down} and the conditioning RMSNorm scale. A directly learned feature scale has no such factorized path. The two parameterizations can therefore represent the same token-invariant rescaling while following different optimization trajectories. If every position is gated, the trained static scale can be folded completely into final RMSNorm; under our convention, preserving the ungated first position requires retaining that one-position exception. Figure 3 shows the impact of rank on NLL with MHAR backbone.

D Efficiency Details

At 115M, LRM adds 66,048 parameters and 131,072 projection FLOPs per token, counting multiplication and addition separately. MHAR adds 25,600 parameters; MHAR + LRM adds 91,648 parameters, or 0.08%, over the Transformer baseline. All methods use one Transformer stack pass. The training-throughput comparison includes forward and backward computation.

Table 8 reports full-context scoring throughput. The exact lookup optimization precomputes the token-dependent rank-space bottleneck but leaves the trained function unchanged.

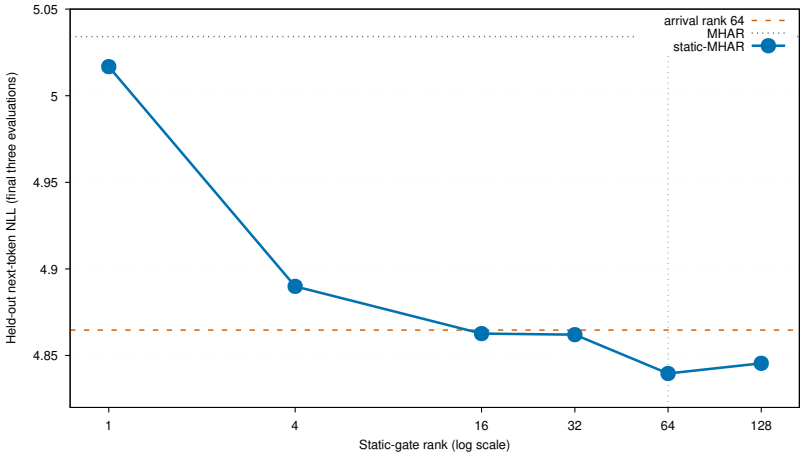

Figure 3: Impact of rank on NLL with MHAR backbone.

Table 8: (MHAR + LRM)/MHAR full-context inference throughput ratios on two A100 80GB GPUs.

Sequence length	Unfused LRM	Exact lookup
128	0.843	0.991
512	0.825	0.993
2,048	0.813	0.993

These measurements score full contexts and are not cached single-token decoding. An isolated readout benchmark measures 0.489–0.504 ms for the gate and a 4.03–4.56 ratio for gate-plus-head relative to the tied head alone; because it omits the Transformer stack, it is an upper bound on readout overhead rather than an end-to-end slowdown. The unfused full-context path is 16–19% slower at batch one, whereas the exact lookup reaches 0.991–0.993 of MHAR throughput.

E Additional Robustness Checks

We run two additional checks to test whether the observed improvement depends on the default precision recipe or can be reproduced by a recurrent input-fusion baseline.

Table 9: Additional matched from-scratch comparisons. Lower NLL is better.

Check	Result
Alternate precision	At 115M and seed 43, with FP32 model state and BF16 autocast, LRM improves final-window NLL by 0.007 on both the Transformer and MHAR backbones.
Compute-matched Full-bandwidth	At 115M and seed 43, the Transformer reaches 3.428 NLL after 12,800 updates, compared with 3.560 for Full-bandwidth after 10,000 updates at equal stack-pass compute.

The alternate-precision result preserves the direction of the main comparison. The Full-bandwidth (Wang et al., 2026) control uses 1.28 passes per update, processes unique tokens at 0.709 times the Transformer rate, and takes 1.102 times as long in the compute-matched comparison. Together, these checks show that LRM's gain persists under a second precision setting and that the recurrent Full-bandwidth baseline does not improve this matched fixed-compute setting.