TRANSFORMING ATTENTION: GENERATING MULTI-HEAD ATTENTION FROM LEARNED GROUP ORBITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-head attention allows transformers to capture different relationships between tokens, but typically learns separate projection parameters for every head. Sharing parameters across heads offers a route to smaller attention blocks, with the central challenge of preserving each head's ability to specialize. We introduce group-transformed multi-head attention (GT-MHA), which generates attention heads by applying learned transformations to a compact set of shared query, key, and value projections. Each head combines shared Lie-algebra generators to form its transformation through the matrix exponential or low-order Taylor approximations. Our analysis shows how heads within a shared orbit represent different interactions within common query and key subspaces. We then extend GT-MHA to multiple learned bases (a multi-orbit formulation), allowing different groups of heads to specialize in different subspaces while controlling the degree of parameter sharing. The resulting architecture preserves the standard multi-head interface and supports key–value caching at the level of shared bases. Under matched BERT pretraining and fine-tuning protocols, residual GT-MHA achieves an official GLUE score of 70.5 versus 70.3 for standard multi-head attention, using 47.2% fewer attention-block parameters. On TinyStoriesV2 character-level language modeling, GT-MHA achieves comparable validation perplexity to standard multi-head attention (1.3331 vs. 1.3355) under the same training budget, with 54.7% fewer attention-block parameters and 75% less persistent KV-cache storage. Ablations and head-diversity analyses provide evidence of learned specialization within shared bases, supporting structured parameter sharing as an effective design for parameter-efficient attention.

1 INTRODUCTION

Multi-head attention (MHA) learns separate query and key projections for each head, giving each head its own way to score relationships between tokens (Vaswani et al., 2017). We can view this head-specific scoring function as a learned bilinear geometry that specifies how query and key coordinates are combined. Yet many heads can be pruned with little loss in performance (Michel et al., 2019; Voita et al., 2019), suggesting that fully independent head parameterizations may offer more flexibility than necessary. But useful differences between heads may not require learning every geometry independently: transformations such as rotations, shears, and stretches can alter how shared query and key representations interact within a common head space, yielding distinct bilinear scoring geometries. This suggests a question: *can a small set of learned base scoring geometries generate diverse, useful heads through structured transformations?*

We introduce Group-Transformed Multi-Head Attention (GT-MHA), which generates attention heads from one or more learned base query, key, and value projections. Shared matrix generators and head-specific coordinates transform each base query–key scoring geometry, while a separate transformation specializes the corresponding value representation. Exact exponentiation places heads on learned group orbits; the residual implementation used in our main experiments provides a first-order approximation. Heads sharing a base vary their bilinear interactions within fixed learned query and key subspaces, whereas multiple bases provide distinct subspaces. The number of bases therefore trades parameter sharing for representational flexibility.

Beyond organizing head diversity, GT-MHA provides parameter and inference-memory savings. Because the base projections are learned once per base rather than once per head, their parameter

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.

count scales with the number of bases instead of the total number of heads. During autoregressive decoding, keys and values can likewise be retained at the base level, so the persistent KV-cache also scales with the number of bases. GT-MHA retains the full collection of attention heads and preserves the standard multi-head input–output interface.

We evaluate GT-MHA in language understanding and autoregressive language modeling under controlled training settings. On BERT-based GLUE evaluation, residual GT-MHA achieves an official score of 70.5, compared with 70.3 for standard MHA, while reducing backbone parameters by 12.2%. On TinyStoriesV2, a 12-layer model using GT-MHA attains a validation perplexity of 1.3331, compared with 1.3355 for MHA at the fixed 250,000-step endpoint, while reducing total model parameters by 18.1%, attention parameters by 54.7%, and persistent KV-cache memory by 75% under base-level caching. These results suggest that structured transformations of shared bases can preserve predictive performance while substantially reducing parameter and inference-memory requirements.

This work makes three contributions:

- We introduce GT-MHA, a structured attention parameterization that generates head-specific scoring geometries and value representations from one or more learned bases through matrix-group transformations.
- We characterize the resulting bilinear scoring forms and their within- and across-orbit structure, analyze parameter, computational, and KV-cache costs, and develop exact and approximate transformation implementations.
- We evaluate GT-MHA in language understanding and autoregressive language modeling, showing comparable predictive performance with substantially fewer parameters and reduced persistent KV-cache memory in autoregressive decoding.

2 Related Work

2.1 Structured Sharing in Multi-Head Attention

Evidence of redundancy and rank constraints in attention heads motivates parameterizations that reduce head-specific structure while retaining multiple attention maps (Michel et al., 2019; Voita et al., 2019; Bhojanapalli et al., 2020). Multi-query attention (MQA) shares one key–value pair across all query heads (Shazeer, 2019), while grouped-query attention (GQA) assigns one pair to each group of query heads (Ainslie et al., 2023). Decoupled-head attention learns layer-dependent groupings for key and value heads when converting pretrained MHA checkpoints (Chen et al., 2024), and multi-head latent attention compresses the cached key–value state through a shared low-rank latent representation (DeepSeek-AI, 2024). These approaches primarily target autoregressive memory and bandwidth through KV sharing or compression. GT-MHA additionally shares query projections within each base and generates head-specific scoring functions through learned transformations of the shared query–key coordinates.

More closely related, Collaborative MHA shares query and key projections across heads and differentiates their scoring functions through head-specific coordinatewise reweighting (Cordonnier et al., 2020). Multiple-head embedding attention uses one seed projection for each of queries, keys, and values, then differentiates heads through additive or elementwise multiplicative embedding vectors (Xue & Aletras, 2023). FiSHformer constructs local attention matrices from a finite admixture of learned global attention matrices (Nguyen et al., 2022). GT-MHA instead applies learned matrix transformations to shared bases, producing orbit-structured heads while allowing multiple bases to provide distinct learned subspaces.

2.2 Group-Based Neural Parameterizations

Group-equivariant models use group actions to encode known data-domain symmetries, including LieConv, group-equivariant stand-alone self-attention, and LieTransformer (Finzi et al., 2020; Romero & Cordonnier, 2021; Hutchinson et al., 2021). GT-MHA uses group structure for a different purpose: the group acts on learned attention parameters rather than on the data domain. Consequently, the construction neither assumes a corresponding symmetry of the data nor guarantees equivariance of

the model's input–output map. Lie-algebraic coordinates and their induced transformations instead organize parameter sharing across heads.

3 BACKGROUND

3.1 MULTI-HEAD ATTENTION AS LEARNED SCORING GEOMETRY

Given an input sequence $X \in \mathbb{R}^{T \times d_{\text{model}}}$, where T denotes the sequence length and d_{model} is the model dimension, each attention head projects the input into query, key, and value spaces:

$$Q_h = XW_Q^{(h)}, \qquad K_h = XW_K^{(h)}, \qquad V_h = XW_V^{(h)},$$

where $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_{\text{model}} \times d_h}$ are learned head-specific projection matrices, d_h is the head dimension, and $h \in \{1, \ldots, H\}$ indexes the H attention heads. Each head forms pairwise query-key scores, normalizes them into attention weights, and uses these weights to aggregate the value representations:

$$\text{Attn}_h(X) = \text{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_h}}\right) V_h.$$

The head outputs are then concatenated and projected back to the model dimension:

$$\text{MHA}(X) = \text{Concat}\left(\text{Attn}_1(X), \ldots, \text{Attn}_H(X)\right) W_O,$$

where $W_O \in \mathbb{R}^{Hd_h \times d_{\text{model}}}$ is the output projection matrix.

The query and key projections induce a bilinear scoring geometry. For token representations $x_i, x_j \in \mathbb{R}^{d_{\text{model}}}$, define

$$B_h = W_Q^{(h)} W_K^{(h)\top} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}, \qquad s_h(x_i, x_j) = \frac{x_i^\top B_h x_j}{\sqrt{d_h}}, \qquad \text{rank}(B_h) \le d_h.$$

We use *scoring geometry* to refer to this bilinear compatibility structure. Note that it need not induce a metric since the corresponding bilinear form may be asymmetric or indefinite.

Standard MHA learns the projection factors of each head independently. Under the common fixed-width choice $d_{\text{model}} = Hd_h$, changing H repartitions a fixed projection dimension into more, narrower heads and leaves the total query, key, and value parameter count unchanged. It therefore changes the number and granularity of independently parameterized scoring geometries, without introducing explicit structure shared among them.

3.2 GROUP ACTIONS AND TRANSFORMATION ORBITS

To relate bilinear scoring geometries through transformations of shared bases, we use group actions and their orbits. A group specifies a family of composable and reversible transformations, an action specifies how those transformations modify a scoring geometry, and an orbit collects the geometries reachable from a given base. Formally, a group G is a set equipped with an associative composition law $(g_1, g_2) \mapsto g_1 g_2$, an identity element e, and an inverse g^{-1} for every $g \in G$. An action of G on a space $\mathcal{S}$ is a map $\Phi : G \times \mathcal{S} \to \mathcal{S}$ satisfying

$$\Phi(e, s) = s, \qquad \Phi(g_1, \Phi(g_2, s)) = \Phi(g_1 g_2, s)$$

for all $g_1, g_2 \in G$ and $s \in \mathcal{S}$. We write $\Phi(g, s) = g \cdot s$. The orbit of $s \in \mathcal{S}$ is $\mathcal{O}(s) = \{g \cdot s : g \in G\}$, the set of objects reachable from s through transformations in G.

A Lie group is both a group and a smooth manifold, with smooth multiplication and inversion. For a matrix Lie group $G \subseteq \text{GL}(d_h, \mathbb{R})$, its Lie algebra $\mathfrak{g} = T_e G$ is the tangent space at the identity and provides a linear space of infinitesimal transformation directions. The exponential map $\exp : \mathfrak{g} \to G$ maps algebra elements to finite group transformations, but need not be surjective.

In the exact construction, GT-MHA takes $\mathcal{S}$ to be a space of bilinear query–key scoring geometries. Let $s^{(0)} \in \mathcal{S}$ denote a learned base geometry. Each learned head-specific element $g_h \in G$ selects a geometry on its orbit $s_h = g_h \cdot s^{(0)} \in \mathcal{O}\big(s^{(0)}\big)$. Using multiple learned bases yields a union of such orbits and allows the degree of sharing across heads to be varied. Here, G acts on the scoring-parameter space $\mathcal{S}$, rather than on the data domain. This action organizes parameter sharing across heads without imposing invariance or equivariance on the model's input–output mapping. Section 4 specifies the matrix action and its parameterization.

4 GROUP-TRANSFORMED MULTI-HEAD ATTENTION (GT-MHA)

GT-MHA generates head-specific query–key scoring geometries from one or more learned bases. We first introduce a Lie-algebraic parameterization that maps head-specific coordinates to transformations of the head space. We then construct single- and multi-orbit variants with different degrees of sharing across heads. The value pathway is parameterized separately, as it determines the content aggregated rather than the query–key compatibility geometry.

4.1 LIE-ALGEBRAIC PARAMETERIZATION OF HEAD TRANSFORMATIONS

GT-MHA constructs its head-specific transformations from a learned generator bank shared across heads. Let $\{E_\ell\}_{\ell=1}^{p} \subset \mathbb{R}^{d_h \times d_h}$ denote the generators, and let $\theta_h \in \mathbb{R}^p$ denote the head-specific logits. We first define

$$\pi_h = \mathrm{softmax}(\theta_h), \qquad A_h = \beta \sum_{\ell=1}^{p} \pi_{h,\ell} E_\ell, \qquad M_h = \exp(A_h).$$

Here, $\pi_{h,\ell} \geq 0$ and $\sum_{\ell=1}^{p} \pi_{h,\ell} = 1$, so π_h determines the contribution of each shared generator to A_h, while $\beta > 0$ controls the overall transformation scale. The generator bank therefore defines transformation directions common to all heads, while θ_h determines how those directions are combined for head h. Thus, M_h is the corresponding head-specific group transformation.

The generators E_ℓ, and hence every A_h, belong to the full matrix Lie algebra $\mathfrak{gl}(d_h, \mathbb{R}) = \mathbb{R}^{d_h \times d_h}$. The matrix exponential maps these elements to the positive-determinant general linear group $\mathrm{GL}^+(d_h, \mathbb{R}) = \left\{M \in \mathbb{R}^{d_h \times d_h} : \det(M) > 0\right\}$. In particular, $\det(M_h) = \exp(\mathrm{tr}(A_h)) > 0$, so the exact transformation is invertible by construction. In our main experiments, we use the first-order residual approximation $M_h^{\mathrm{res}} = I + A_h$. This retains the shared-generator and shared-base structure but does not guarantee invertibility or exact group membership; Figure 3(c) compares its predictive performance with second-order approximation and exact exponentiation.

4.2 SINGLE-ORBIT GT-MHA

We introduce GT-MHA through its most strongly shared case, the single-orbit construction, in which all H attention heads are generated from one learned base query–key geometry. Shared query and key projections define this base, while M_h from Section 4.1 determines each head's scoring geometry within the common d_h-dimensional head space. Thus, head diversity is expressed through M_h, rather than independently learned query–key projection pairs as in standard MHA.

Let $W_Q^{(0)}, W_K^{(0)} \in \mathbb{R}^{d_{\mathrm{model}} \times d_h}$ denote the shared base query and key projections. For each head h, M_h acts within this shared head space, inducing the effective scoring matrix and pre-softmax attention scores

$$B_h := W_Q^{(0)} M_h W_K^{(0)\top}, \quad S_h := \frac{X B_h X^\top}{\sqrt{d_h}} = \frac{Q^{(0)} M_h K^{(0)\top}}{\sqrt{d_h}} \in \mathbb{R}^{T \times T}.$$

The identity $M = I$ recovers the base scoring matrix $B^{(0)} = W_Q^{(0)} W_K^{(0)\top}$. Varying M over $G = \mathrm{GL}^+(d_h, \mathbb{R})$ defines the ambient scoring orbit $\mathcal{O}_{QK}^{(0)} = \left\{W_Q^{(0)} M W_K^{(0)\top} : M \in G\right\}$.

Since $M_h = \exp(A_h) \in G$, each B_h lies on this common orbit although the learned generator parameterization generally realizes only a subset of it. The shared projections determine the query and key subspaces, while M_h determines the bilinear interaction within those subspaces.

Proposition 1 (Shared-subspace structure) *Suppose $W_Q^{(0)}$ and $W_K^{(0)}$ have full column rank. For any $M \in \mathbb{R}^{d_h \times d_h}$, the linear map $M \mapsto W_Q^{(0)} M W_K^{(0)\top}$ is injective and rank-preserving. Its image consists precisely of the scoring matrices whose column space lies in $\mathrm{col}(W_Q^{(0)})$ and whose transpose has column space lying in $\mathrm{col}(W_K^{(0)})$. Consequently, $M_h = \exp(A_h)$ implies $\mathrm{rank}(B_h) = d_h$.*

Appendix B provides a proof. Restricting M to $\mathrm{GL}^+(d_h, \mathbb{R})$ yields the ambient scoring orbit: the base projections fix the query and key subspaces, while M_h selects a full-rank bilinear interaction

within them. Injectivity ensures that distinct transformations produce distinct scoring matrices, although diversity across heads is learned rather than imposed.

While S_h determines where head h attends, the value pathway determines what it aggregates. Let $W_V^{(0)} \in \mathbb{R}^{d_{\text{model}} \times d_h}$, $V^{(0)} = XW_V^{(0)}$, and let $\{E_\ell^V\}_{\ell=1}^{p_V} \subset \mathbb{R}^{d_h \times d_h}$ and $\theta_h^V \in \mathbb{R}^{p_V}$ denote the shared value generators and head-specific logits. Analogously to the query–key pathway, we define

$$\pi_h^V = \mathrm{softmax}(\theta_h^V), \qquad A_h^V = \beta_V \sum_{\ell=1}^{p_V} \pi_{h,\ell}^V E_\ell^V, \qquad R_h^V = \exp(A_h^V), \qquad V_h = V^{(0)} R_h^V.$$

The residual implementation analogously replaces R_h^V by $I + A_h^V$. Because the value generator bank is learned independently of the query–key bank, R_h^V changes the aggregated content without affecting the attention weights. With $P_h = \mathrm{softmax}(S_h)$,

$$Z_h = P_h V_h, \qquad \text{GT-MHA}(X) = \mathrm{Concat}(Z_1, \ldots, Z_H) W_O.$$

Thus, GT-MHA preserves the standard aggregation and output interface while generating its scoring forms and value representations from shared bases.

The value transformation can also be viewed as a reparameterization of the output projection. Let $W_{O,h}$ denote the block of W_O applied to head h. Then

$$P_h V^{(0)} R_h^V W_{O,h} = P_h V^{(0)} \widetilde{W}_{O,h}, \qquad \widetilde{W}_{O,h} := R_h^V W_{O,h}.$$

Thus, along the linear value-to-output pathway, R_h^V and $W_{O,h}$ can be combined into a single effective output projection. When R_h^V is invertible, as in the exact construction, this correspondence is one-to-one, so the value transformation does not change the function class of the shared-value model. Its potential benefit instead comes from the structured training parameterization it induces, which may affect optimization and regularization rather than inference expressivity. We evaluate the empirical effect of the value transformation in Section 6.4 (Table 5(a)).

4.3 Multi-Orbit GT-MHA

Multi-orbit GT-MHA increases subspace-level expressivity by allowing different groups of heads to operate in distinct query, key, and value subspaces. It learns base projections $W_Q^{(c)}, W_K^{(c)}, W_V^{(c)}$ for $c \in \{0, \ldots, C-1\}$ and assigns each head h to a fixed base $c_h \in \{0, \ldots, C-1\}$, retaining structured sharing among heads assigned to the same base. We adopt balanced assignments with H/C heads per base in our experiments, although the construction permits unequal group sizes.

For a head h assigned to base c_h, let $Q^{(c_h)} = XW_Q^{(c_h)}$ and $K^{(c_h)} = XW_K^{(c_h)}$. Its effective scoring matrix and pre-softmax attention scores are defined using the transformation M_h as

$$B_h := W_Q^{(c_h)} M_h W_K^{(c_h)\top}, \quad S_h := \frac{XB_hX^\top}{\sqrt{d_h}} = \frac{Q^{(c_h)} M_h K^{(c_h)\top}}{\sqrt{d_h}} \in \mathbb{R}^{T \times T}.$$

The generator bank remains shared across all heads and bases, while the coordinates θ_h remain head-specific. The multi-orbit extension changes only the base projections assigned to each head. Each base projection pair defines an ambient scoring orbit $\mathcal{O}_{QK}^{(c)} = \left\{ W_Q^{(c)} M W_K^{(c)\top} : M \in G \right\}$. Consequently, the scoring matrix $B_h \in \mathcal{O}_{QK}^{(c_h)}$, so the head-specific scoring matrices belong to the union of these C orbits: $\{B_h\}_{h=1}^H \subseteq \bigcup_{c=0}^{C-1} \mathcal{O}_{QK}^{(c)}$.

Multi-orbit GT-MHA distributes heads across learned subsets of these ambient orbits. Proposition 1 applies separately to each base whose query and key projections have full column rank. Thus, transformations within an orbit vary bilinear interactions inside fixed base subspaces, whereas different orbits may provide different query and key subspaces. This across-orbit variation supplies the additional expressivity of the multi-orbit construction.

The value pathway follows the same base assignment. Using the independently learned value transformation R_h^V, head h forms

$$V_h = XW_V^{(c_h)} R_h^V, \qquad Z_h = \mathrm{softmax}(S_h) V_h.$$

The value generator bank also remains shared across all heads and bases. The head outputs are concatenated and mixed by W_O as in the single-orbit construction. Appendix C.5 provides the scoring schematic (Figure 4) and forward-pass pseudocode (Algorithm 1).

5 ANALYSIS

GT-MHA decouples the number of attention heads H from the number of independently learned projection triples C. Heads sharing a base can learn different scoring geometries through their transformations. The number of bases C controls the degree of sharing and the number of independently learned query and key subspace pairs available to the heads. The choice of C therefore controls subspace-level expressivity; its effect on task performance is evaluated empirically in 6.4.

With p query–key and p_V value generators, the Q/K/V parameter count, excluding biases and the unchanged output projection, becomes $3Cd_{\text{model}}d_h + (p+p_V)(d_h^2+H)$, compared with $3Hd_{\text{model}}d_h$ for MHA. Note that these savings reflect the reduction in Q/K/V parameters after accounting for the generator banks and head-specific coordinates; the corresponding whole-network reduction depends on the proportion of parameters allocated to these components. GT-MHA retains H attention maps and the $O(HT^2d_h)$ attention cost, while reducing base-projection work and adding transformation overhead. During autoregressive decoding, storing only the base keys and values reduces persistent KV-cache memory to C/H of MHA. For example, with $H = 16$, $C = 4$, $d_h = 64$, and BF16, this is 1024 rather than 4096 bytes per token per layer. Appendix C derives these counts and separates analytical savings from implementation-dependent runtime costs.

6 EXPERIMENTS

We evaluate whether GT-MHA preserves predictive performance while reducing attention parameters on BERT+GLUE and TinyStoriesV2. Controlled ablations isolate the effects of learned transformations, base count, and transformation mapping.

6.1 EXPERIMENTAL SETUP

Comparison protocol. We compare GT-MHA with MHA, GQA, and Collaborative MHA and additionally include MQA for autoregressive modeling. Unless stated otherwise, GT-MHA uses the first-order residual mapping $I + A_h$. Second-order and exact-exponential variants are evaluated only in Section 6.4. Within each experimental setting, we hold fixed the non-attention architecture, training data and order, optimization schedule, and training budget across methods. Appendix D provides the complete configurations and checkpoint protocols.

BERT and GLUE. We pretrain BERT-base–scale encoders (Devlin et al., 2019) from random initialization on WikiText-103 (Merity et al., 2016) for 100,000 masked-language-modeling updates using three matched random seeds. For each method, we select the pretrained encoder with the lowest MLM validation loss across these seeds and fine-tune it on all nine GLUE tasks (Wang et al., 2018). For each downstream task, we select the learning rate with the highest mean validation score over three initial fine-tuning seeds. We then run two additional seeds at the selected rate and submit the checkpoint with the highest validation score among the five runs to the official GLUE test server. Neither learning-rate nor checkpoint selection uses test labels. All reported GLUE results, including the main comparison, transformation ablation, and base-count ablation, are official test-server scores. Appendix D provides the complete configurations and selection procedure.

TinyStories. We train 12-layer decoder-only language models on TinyStoriesV2 (Eldan & Li, 2023) using character-level tokenization. We report token-weighted validation negative log-likelihood and perplexity and assess generation quality through blinded GPT-4.1 judgments under identical prompts and decoding settings. Each method uses one training run. For the likelihood comparison, paired bootstrap resampling of validation stories quantifies evaluation-set uncertainty but not variability across independent training runs. Appendix H details the fixed-checkpoint evaluation and bootstrap procedures. Appendix D provides the training configurations, and Appendix I describes the generation and grading protocol.

6.2 NATURAL LANGUAGE UNDERSTANDING

Table 1 reports the official GLUE test results. GT-MHA attains the highest observed aggregate score, 70.5, while using the smallest backbone and the fewest attention parameters among the

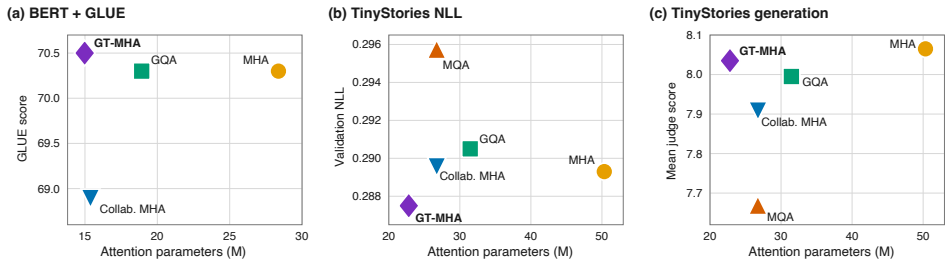

Figure 1: **Parameter–performance trade-offs.** Lower attention parameter counts and TinyStoriesV2 NLL are better. Higher GLUE and generation scores are better.

Table 1: **Official GLUE test results.** Backbone counts exclude task-specific heads, and Attn. counts include all attention-block parameters. Slash-separated entries follow GLUE conventions.

Method	Backbone (M)	Attn. (M)	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	GLUE
MHA	109.5	28.37	28.0	88.2	**81.9**/73.7	**78.4/76.9**	68.1/**87.5**	**77.7**/76.7	86.1	54.7	**65.1**	70.3
GQA	100.0	18.92	29.0	87.4	81.4/**74.6**	76.3/74.6	67.9/87.4	76.9/75.8	85.5	**58.2**	**65.1**	70.3
Collaborative MHA	96.5	15.38	30.8	88.4	78.0/68.0	73.1/71.2	66.7/86.9	75.4/74.4	85.5	53.7	**65.1**	68.9
GT-MHA (Ours)	96.1	14.98	**31.5**	**89.2**	79.1/69.7	77.5/76.0	**68.3/87.5**	**77.7/76.9**	**86.3**	56.3	**65.1**	**70.5**

evaluated methods. MHA and GQA both score 70.3 with 109.5M and 100.0M backbone parameters, respectively, compared with 96.1M for GT-MHA. Relative to MHA, GT-MHA reduces attention parameters by 47.2% and backbone parameters by 12.2%. At a similar backbone size, Collaborative MHA scores 68.9 with 96.5M parameters, 1.6 points below GT-MHA.

Task-level performance is mixed. GT-MHA is best or tied for the best result on CoLA, SST-2, QQP, MNLI, QNLI, and WNLI, including gains of 3.5 and 1.0 points over MHA on CoLA and SST-2, respectively. MHA retains an advantage on MRPC and STS-B, while GQA performs best on RTE. Thus, GT-MHA provides the strongest observed aggregate parameter–performance trade-off, although its gains are not uniform across tasks. Appendix G reports validation means and standard deviations across fine-tuning seeds at the selected learning rates.

6.3 Language Modeling and Generation

GT-MHA achieves the lowest observed language-modeling loss among the evaluated checkpoints while using the fewest parameters (Table 2). On the full TinyStoriesV2 validation set, it obtains a NLL of 0.2875 and perplexity of 1.3331, compared with 0.2893 and 1.3355 for MHA while reducing attention parameters from 50.33M to 22.81M (54.7%) and total parameters from 152.10M to 124.58M (18.1%). GQA, MQA, and Collaborative MHA obtain perplexities of 1.3370, 1.3441, and 1.3359, respectively, while using more parameters than GT-MHA. Thus, among the evaluated configurations, GT-MHA provides the best trade-off between model size and language-modeling performance.

Figure 7 summarizes the paired-bootstrap comparisons for language modeling and generation. Resampling the TinyStories validation stories yields an NLL difference of -0.00177 between GT-MHA and MHA, with a 95% confidence interval of $[-0.00191, -0.00163]$. Because the interval lies entirely below zero, the observed NLL advantage is consistent under resampling of the evaluation stories.

For generation, GT-MHA obtains an average rating of 8.035, compared with 8.065 for MHA, while achieving the highest observed creativity score (6.981). The paired average-score difference is -0.0303, with a 95% prompt-bootstrap interval of $[-0.1122, +0.0508]$. The interval does not support a generation-quality advantage for either model. Nevertheless, GT-MHA attains a closely matched mean score while using 54.7% fewer attention parameters than MHA. Relative to the other parameter-sharing baselines, GT-MHA's mean generation score is higher by 0.040 over GQA, 0.125 over Collaborative MHA, and 0.369 over MQA. Paired-bootstrap intervals provide evidence for the advantages over Collaborative MHA and MQA, whereas the smaller difference from GQA remains unresolved by this evaluation. Note that these intervals capture prompt-sampling uncertainty, not

Table 2: **TinyStoriesV2 results.** Full-validation language-modeling metrics and mean blinded GPT-4.1 ratings over 1,000 generations per method.

	(a) Language modeling				(b) Generation quality				
Method	**Total (M)**	**Attn. (M)**	**NLL ↓**	**PPL ↓**	**Grammar ↑**	**Consistency ↑**	**Creativity ↑**	**Plot ↑**	**Average ↑**
MHA	152.10	50.33	0.2893	1.3355	**8.743**	**8.568**	6.950	**8.000**	**8.065**
GQA (4 KV heads)	133.23	31.46	0.2905	1.3370	8.663	8.434	6.959	7.924	7.995
MQA	128.51	26.74	0.2957	1.3441	8.230	8.007	6.880	7.549	7.667
Collaborative MHA	128.52	26.75	0.2896	1.3359	8.524	8.307	6.957	7.851	7.910
GT-MHA (Ours)	124.58	22.81	**0.2875**	**1.3331**	8.707	8.509	**6.981**	7.943	8.035

variability across training runs. Appendix H details the full-validation evaluation and story-bootstrap procedure, while Appendix I provides the generation protocol and complete pairwise intervals.

6.4 Structural Ablations

We use controlled ablations to answer three questions: whether the learned query–key and value transformations contribute beyond base sharing, how the number of bases controls the compression–performance trade-off, and whether exact exponentiation is required for predictive performance. Identity comparisons use $C = 4$ and otherwise match the corresponding GT-MHA configurations. BERT results use the official GLUE test server under the selection protocol of Section 6.1, whereas TinyStories NLL uses one run per configuration and the fixed-checkpoint, full-validation protocol in Appendix H. Complete numerical results are provided in Appendix E.

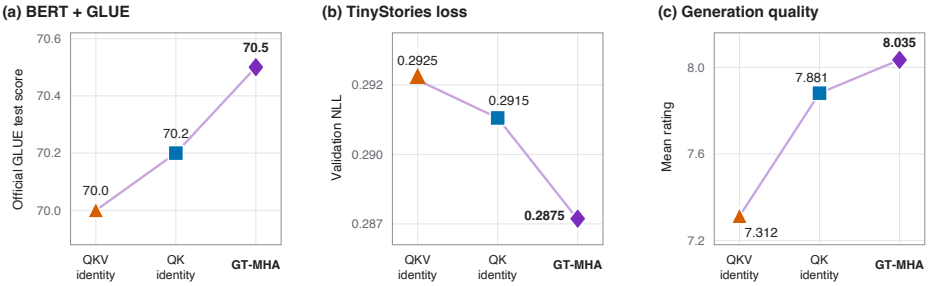

Figure 2: **Contribution of learned transformations.** QKV identity removes both transformations, QK identity restores the value pathway, and GT-MHA restores all transformations. Both additions improve GLUE and generation quality while reducing TinyStories NLL.

Contribution of transformations. Figure 2 evaluates a sequence of models. QKV identity removes transformation pathways, QK identity restores the value transformation while keeping the scoring transformation fixed to the identity, and GT-MHA enables both pathways. This design separates the contributions of transforming the aggregated content and the query–key scoring geometry.

Both transformations improve the observed results. Restoring the value transformation improves TinyStories validation NLL from 0.2925 to 0.2915, generation quality from 7.312 to 7.881, and the official GLUE score from 70.0 to 70.2. Enabling the query–key transformation further improves these metrics to 0.2875, 8.035, and 70.5, respectively. Thus, the value pathway produces the larger generation-quality gain, while the learned scoring transformation produces the larger reduction in language-modeling loss. This pattern is consistent with their architectural roles: the query–key transformation changes where and how strongly a head attends, whereas the value transformation leaves the attention weights unchanged and changes the representations being aggregated.

For generation, the paired difference between GT-MHA and QK identity is $+0.1538$, with a 95% prompt-bootstrap interval of $[+0.0657, +0.2390]$. Together with the likelihood results, this supports a contribution from head-specific scoring transformations beyond base sharing. Appendix F.2 shows modest within-base specialization in BERT and stronger specialization in TinyStories.

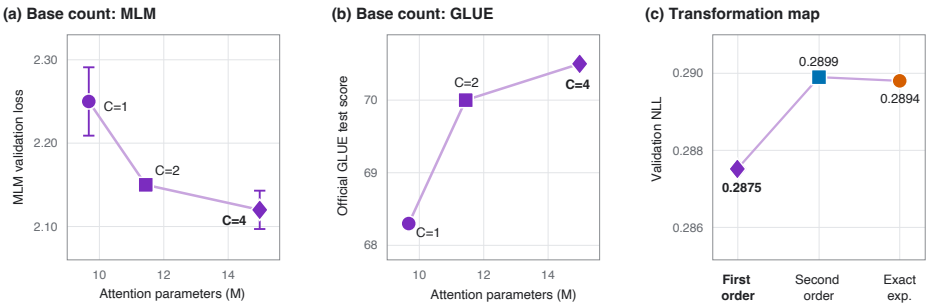

Figure 3: **Base-count and transformation-mapping ablations.** (a,b) Increasing C improves MLM loss and official GLUE performance; MLM error bars show standard deviations over three pretraining seeds. (c) The first-order mapping achieves the lowest TinyStories validation NLL.

Number of bases. Increasing C relaxes projection sharing by providing more independently learned base subspaces (Figures 3(a,b)). As C increases from 1 to 2 to 4, mean MLM validation loss decreases from 2.25 to 2.15 to 2.12, and the official GLUE test score increases from 68.3 to 70.0 to 70.5, while attention parameters increase from 9.67M to 11.44M to 14.98M. Most of the performance gain occurs between $C = 1$ and $C = 2$, whereas $C = 4$ provides a smaller additional gain, indicating diminishing returns from additional bases. These results validate C as a direct control over the expressivity–compression trade-off and motivate $C = 4$ as our operating point.

Transformation mapping. First-order residual mapping achieves the lowest TinyStories validation NLL, 0.2875, compared with 0.2899 for the second-order approximation and 0.2894 for exact exponentiation (Figure 3(c)). Greater fidelity to the matrix exponential therefore does not improve predictive performance in these runs. One interpretation is that the first-order mapping provides a useful structural relaxation rather than merely a computational approximation: it preserves the shared-base and shared-generator structure while dropping exact group membership and making M_h affine in A_h. This may ease optimization or admit useful transformations outside the exponential image. Appendix F analyzes transformation conditioning, distance from identity, and functional head diversity, while Appendix D reports the matched efficiency measurements.

7 Limitations and Future Directions

Adaptive sharing. GT-MHA assigns each head to a fixed base, constraining heads within a group to shared query and key subspaces and all heads to a common generator bank. Learning these assignments or allowing mixtures of bases could increase flexibility while preserving structured sharing. **Efficient implementation.** Matched GH200 measurements confirm that GT-MHA reduces operator-reported computation by 12.6%, peak allocated training memory by 6.5%, and persistent TinyStories KV-cache storage by 75% relative to MHA. The current unfused implementation attains lower throughput, highlighting a clear opportunity for specialized fused kernels to realize these structural savings in wall-clock performance. **Evaluation scope.** We evaluate GT-MHA on BERT-based language understanding and TinyStories language modeling, leaving larger scales and other modalities unexplored. TinyStories comparisons use one training run per variant, so training-seed variability remains uncharacterized.

8 Conclusion

We introduced group-transformed multi-head attention (GT-MHA), which generates head-specific scoring functions and value representations through learned transformations of shared base projections. The single-orbit construction provides maximal sharing, while the multi-orbit extension enables specialization across distinct learned subspaces. Across BERT-based language understanding and TinyStoriesV2 language modeling, GT-MHA achieves comparable predictive performance with substantially fewer attention-block parameters and reduces persistent KV-cache storage during autoregressive decoding. Ablations and head-diversity analyses support the contribution of learned transformations within shared bases. These results demonstrate that structured sharing retains head specialization while reducing the parameter and storage costs of multi-head attention.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Low-rank bottleneck in multi-head attention models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 864–873, 2020.

Yilong Chen, Linhao Zhang, Junyuan Shang, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, and Yu Sun. Dha: Learning decoupled-head attention from transformer checkpoints via adaptive heads fusion. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*, 2020.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3165–3176, 2020.

Michael J. Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4533–4543, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Tan Nguyen, Tam Nguyen, Hai Do, Khai Nguyen, Vishwanath Saragadam, Minh Pham, Khuong Duy Nguyen, Nhat Ho, and Stanley Osher. Improving transformer with an admixture of attention heads. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

David W. Romero and Jean-Baptiste Cordonnier. Group equivariant stand-alone self-attention for vision. In *International Conference on Learning Representations*, 2021.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Max Staats, Matthias Thamm, and Bernd Rosenow. Small singular values matter: A random matrix analysis of transformer models. *Advances in Neural Information Processing Systems*, 38: 153545–153573, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Huiyin Xue and Nikolaos Aletras. Pit one against many: Leveraging attention-head embeddings for parameter-efficient multi-head attention. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10355–10373, 2023.

A Large Language Model Usage

OpenAI Codex was used for language editing, manuscript organization, literature discovery, and assistance with coding and debugging. The research ideas, proposed method, theoretical development, experimental design, results, and conclusions are the authors' work. All LLM-assisted material was reviewed and verified by the authors, who take full responsibility for the contents of the paper.

B Proof of Proposition 1

Proof. Let $U = W_Q^{(0)}$ and $V = W_K^{(0)}$. Because U and V have full column rank, their left pseudoinverses

$$U^\dagger = (U^\top U)^{-1}U^\top, \qquad V^\dagger = (V^\top V)^{-1}V^\top$$

satisfy $U^\dagger U = V^\dagger V = I_{d_h}$. For any $M \in \mathbb{R}^{d_h \times d_h}$,

$$\mathrm{col}(UMV^\top) \subseteq \mathrm{col}(U), \qquad \mathrm{col}\big((UMV^\top)^\top\big) \subseteq \mathrm{col}(V),$$

so every matrix in the image satisfies the stated subspace constraints.

Conversely, let $B \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$ satisfy

$$\mathrm{col}(B) \subseteq \mathrm{col}(U), \qquad \mathrm{col}(B^\top) \subseteq \mathrm{col}(V).$$

Define $P_U = UU^\dagger$ and $P_V = VV^\dagger$, which are the symmetric orthogonal projectors onto $\mathrm{col}(U)$ and $\mathrm{col}(V)$, respectively. These conditions imply $P_U B = B$ and $BP_V = B$. Setting

$$M = U^\dagger B (V^\dagger)^\top$$

therefore gives

$$UMV^\top = P_U B P_V = B.$$

Hence, the image is precisely the claimed family of scoring matrices.

Moreover,

$$U^\dagger (UMV^\top)(V^\dagger)^\top = M.$$

Thus $M \mapsto UMV^\top$ is injective. The general rank inequality gives $\mathrm{rank}(UMV^\top) \le \mathrm{rank}(M)$, while the displayed recovery of M gives the reverse inequality. Therefore,

$$\mathrm{rank}(UMV^\top) = \mathrm{rank}(M).$$

Finally, under the exact parameterization, $M_h = \exp(A_h)$ is invertible, so $\mathrm{rank}(M_h) = d_h$. Taking $M = M_h$ therefore yields $\mathrm{rank}(B_h) = d_h$. ∎

C Detailed Analytical Properties

This section expands Section 5, distinguishing projection sharing, attention-module parameter savings, forward-pass computation, and persistent decoding memory. Throughout, H is the number of heads, C the number of base projection triples, d_h the head width, and d_{model} the model width.

C.1 Sharing and Expressivity

The number of bases C controls how much independently learned projection capacity GT-MHA provides. With $C = 1$, all heads derive from one base projection triple; with $C = H$ and identity transformations, the construction recovers the per-head projection structure of standard MHA. Within a base, head-specific transformations can produce different scoring geometries and value mappings while retaining shared projections. Additional bases allow heads to draw on distinct learned query and key subspaces: transformations within a single base cannot introduce directions outside that base's subspaces.

The effect of C on predictive performance is examined in Section 6.4, with results in Figure 3(a,b) and Table 5(b). The largest improvement occurs when moving from one to two bases, suggesting that a modest increase in independently learned projection capacity can be useful while preserving substantial sharing.

C.2 Parameter Count

Let p and p_V denote the numbers of query–key and value generators. Omitting biases, the base projections, shared generator banks, and head-specific coordinates contain

$$P_{\mathrm{QKV}}^{\mathrm{GT}} = 3Cd_{\mathrm{model}}d_h + (p + p_V)(d_h^2 + H), \qquad P_{\mathrm{QKV}}^{\mathrm{MHA}} = 3Hd_{\mathrm{model}}d_h.$$

Both methods retain an output projection with $Hd_{\mathrm{model}}d_h$ parameters. The complete attention-module counts are therefore

$$\begin{aligned} P_{\mathrm{attn}}^{\mathrm{GT}} &= (3C + H)d_{\mathrm{model}}d_h + (p + p_V)(d_h^2 + H), \\ P_{\mathrm{attn}}^{\mathrm{MHA}} &= 4Hd_{\mathrm{model}}d_h. \end{aligned}$$

A net reduction occurs when

$$(p + p_V)(d_h^2 + H) < 3(H - C)d_{\mathrm{model}}d_h.$$

In particular, the fractional reduction in complete attention-module parameters is

$$1 - \frac{P_{\mathrm{attn}}^{\mathrm{GT}}}{P_{\mathrm{attn}}^{\mathrm{MHA}}} = \frac{3}{4}\left(1 - \frac{C}{H}\right) - \frac{(p + p_V)(d_h^2 + H)}{4Hd_{\mathrm{model}}d_h}.$$

The factor H/C applies to the base Q/K/V projection weights; the output projection remains unchanged, and the generator banks and head-specific coordinates add a smaller parameter cost. The measured models show the resulting savings. With $C = 4$, BERT attention-block parameters fall from 28.37M for MHA to 14.98M for GT-MHA (47.2%), while backbone parameters fall from 109.5M to 96.1M (12.2%; Table 1). For TinyStories, attention-block parameters fall from 50.33M to 22.81M (54.7%), and total parameters from 152.10M to 124.58M (18.1%; Table 2). Table 5(b) further shows how the attention parameter count changes with C.

C.3 Computational Complexity

Table 3 gives per-layer forward-pass costs for a sequence of length T, suppressing the batch dimension and constant factors. The comparison assumes dense generators and matched H, d_h, and d_{model}. GT-MHA reduces base Q/K/V projection work from $O(HTd_{\mathrm{model}}d_h)$ to $O(CTd_{\mathrm{model}}d_h)$. Combining generators and applying the resulting head transformations introduce additional work, while attention-score computation, value aggregation, and the output projection retain their MHA scaling.

For a head-specific generator combination A_h, the exact construction uses $M_h = \exp(A_h)$, whereas the residual formulation uses $M_h = I + A_h$; the same distinction applies to the value transformation. After generator combination, forming the residual transformation costs $O(Hd_h^2)$. Dense matrix exponentiation has a conventional $O(Hd_h^3)$ cost, with constants depending on the numerical algorithm. Exact exponentiation guarantees membership in $\mathrm{GL}^+(d_h, \mathbb{R})$; the residual formulation is not guaranteed to be invertible and does not inherit this guarantee.

At inference, head transformations depend only on fixed model parameters and can be precomputed rather than reconstructed for every token. Both formulations retain the $O(HT^2d_h)$ attention cost. In the matched BERT benchmark, the PyTorch profiler counts 5.90 trillion floating-point operations

Table 3: Per-layer forward-pass complexity. Residual GT-MHA uses $I + A_h$; exact GT-MHA uses $\exp(A_h)$. Costs omit the batch dimension and constant factors.

Operation	**MHA**	**GT-MHA (residual)**	**GT-MHA (exact)**
Base Q/K/V projections	$O(HTd_{\text{model}}d_h)$	$O(CTd_{\text{model}}d_h)$	$O(CTd_{\text{model}}d_h)$
Generator combination	–	$O(H(p+p_V)d_h^2)$	$O(H(p+p_V)d_h^2)$
Transformation evaluation	–	$O(Hd_h^2)$	$O(Hd_h^3)$
Applying transformations	–	$O(HTd_h^2)$	$O(HTd_h^2)$
Attention and aggregation	$O(HT^2d_h)$	$O(HT^2d_h)$	$O(HT^2d_h)$
Output projection	$O(HTd_{\text{model}}d_h)$	$O(HTd_{\text{model}}d_h)$	$O(HTd_{\text{model}}d_h)$

(TFLOPs) per optimizer step for residual GT-MHA, compared with 6.75 trillion for MHA—a 12.6% lower reported count (Appendix D). The lower reported count is consistent with reduced base-projection work, although the profiler omits unsupported fused or custom operations. The current unfused GT-MHA implementation nevertheless has lower training throughput, motivating specialized kernels to translate these structural savings into speed. The TinyStories mapping ablation further shows higher throughput for the residual mapping than for exact exponentiation (329.5 versus 306.1k tokens/s/GPU; Table 5(c)), consistent with its lower transformation cost.

C.4 Base-Level KV Caching

During autoregressive decoding, only the keys and values of previous tokens must persist; the current query is consumed immediately. For batch size n, cached length T, and b bytes per scalar, standard MHA stores

$$K_{\text{MHA}}, V_{\text{MHA}} \in \mathbb{R}^{n\times H\times T\times d_h}, \qquad \mathcal{M}_{\text{KV}}^{\text{MHA}} = 2nTHd_hb.$$

For GT-MHA, let $K^{(c)}, V^{(c)} \in \mathbb{R}^{n\times T\times d_h}$ be the streams produced by base c. A new query can score directly against its cached base keys:

$$S_h = \frac{\big(q^{(c_h)}M_h\big)K^{(c_h)\top}}{\sqrt{d_h}}.$$

No head-specific key history is required. Linearity also allows the value transformation to follow aggregation:

$$\text{softmax}(S_h)\big(V^{(c_h)}R_h^V\big) = \big(\text{softmax}(S_h)V^{(c_h)}\big)R_h^V.$$

Consequently, a base-cache implementation retains only

$$K_{\text{GT}}, V_{\text{GT}} \in \mathbb{R}^{n\times C\times T\times d_h}, \qquad \mathcal{M}_{\text{KV}}^{\text{GT}} = 2nTCd_hb = \frac{C}{H}\mathcal{M}_{\text{KV}}^{\text{MHA}}.$$

For unequal key and value widths, the corresponding count is $nTC(d_k+d_v)b$. With $H=16$, $C=4$, $d_h=64$, and BF16 storage ($b=2$), persistent KV storage falls by 75%, from 4096 to 1024 bytes per token per layer.

The TinyStories decoder confirms this analytical count. At batch size one, measured cache-tensor storage across 12 layers is 12,288 bytes per cached token for GT-MHA, versus 49,152 bytes for MHA (Appendix D). This is the same 75% persistent KV-cache reduction as GQA with four KV heads, while GT-MHA retains H head-specific attention distributions. The reduction applies to persistent KV storage; total inference memory and latency also depend on temporary tensors and kernel implementation.

C.5 GT-MHA Forward Pass

Figure 4 illustrates the query–key construction, while Algorithm 1 summarizes the complete attention computation for a single layer. It uses learned base projections $\{W_Q^{(c)}, W_K^{(c)}, W_V^{(c)}\}_{c=0}^{C-1}$, output projection W_O, shared generator banks $\{E_\ell\}_{\ell=1}^{p}$ and $\{E_\ell^V\}_{\ell=1}^{p_V}$, and head-specific logits θ_h, θ_h^V.

The fixed assignment c_h selects the base for head h; $C = 1$ recovers single-orbit GT-MHA. The transformation map is chosen as

$$f(A) = \begin{cases} \exp(A), & \text{exact,} \\ I + A, & \text{first-order (default),} \\ I + A + \frac{1}{2}A^2, & \text{second-order,} \end{cases}$$

where exp denotes the matrix exponential and $A^2 = AA$. The additive attention mask Ω denotes the causal or padding mask, as appropriate. Batch dimensions, projection biases, and dropout are omitted for clarity.

Algorithm 1 GT-MHA forward pass with C shared bases and H heads.

Require: Input $X \in \mathbb{R}^{T \times d_{\text{model}}}$; mask $\Omega \in \mathbb{R}^{T \times T}$
Require: Base assignments $\{c_h\}_{h=1}^{H}$; scales β, β_V; transformation map f
for $c = 0, \ldots, C-1$ **do**
  $Q^{(c)} \leftarrow XW_Q^{(c)}, \quad K^{(c)} \leftarrow XW_K^{(c)}, \quad V^{(c)} \leftarrow XW_V^{(c)}$
end for
for $h = 1, \ldots, H$ **do**
  $\pi_h \leftarrow \text{softmax}(\theta_h), \quad \pi_h^V \leftarrow \text{softmax}(\theta_h^V)$
  $A_h \leftarrow \beta \sum_{\ell=1}^{p} \pi_{h,\ell} E_\ell, \quad A_h^V \leftarrow \beta_V \sum_{\ell=1}^{p_V} \pi_{h,\ell}^V E_\ell^V$
  $M_h \leftarrow f(A_h), \quad R_h^V \leftarrow f(A_h^V)$
  $S_h \leftarrow (Q^{(c_h)} M_h) K^{(c_h)\top} / \sqrt{d_h}$
  $P_h \leftarrow \text{softmax}_{\text{row}}(S_h + \Omega)$
  $Z_h \leftarrow (P_h V^{(c_h)}) R_h^V$
end for
return $\text{Concat}(Z_1, \ldots, Z_H) W_O$

The value transformation is applied after aggregation using the equivalent formulation in Appendix C.4; autoregressive decoding can therefore retain keys and values at the base level.

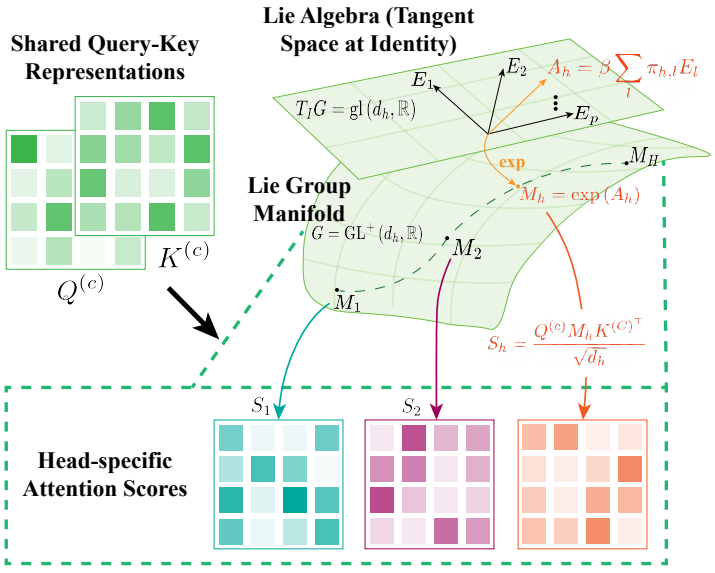

Figure 4: Schematic of the GT-MHA query–key construction for heads sharing base c. Head-specific mixtures of shared generators define A_h; exact matrix exponentiation maps A_h to $M_h \in \mathrm{GL}^+(d_h, \mathbb{R})$. Combining M_h with the shared query and key representations yields scores S_h. The surface schematically depicts the matrix Lie group, and the plane depicts its tangent space at the identity, identified with the Lie algebra.

Table 4: Training settings. Global batch sizes count sequences per optimizer update, including gradient accumulation. TinyStories uses character tokens. Transformation scales and generator counts apply to GT-MHA.

Setting	BERT pretraining	GLUE fine-tuning	TinyStories
Tokenizer	BERT uncased WordPiece	BERT uncased WordPiece	Character-level
Metric Beta (β)	1.0	1.0	1.0
Value Beta (β_V)	1.0	1.0	1.0
Metric Generators (p)	8	8	8
Value Generators (p_V)	8	8	8
Layers / hidden width	12 / 768	12 / 768	12 / 1024
Nominal attention heads	12	12	16
Maximum sequence length	128	128	512
Global training batch	256	128 (except WNLI)	2048
Optimizer	AdamW	AdamW	AdamW
Learning rate	5×10^{-5}	$\{2, 3, 5\} \times 10^{-5}$	3×10^{-4}
Warm-up	10,000 steps	10% of training steps	None
LR schedule	Linear decay	Linear decay	Hold, then cosine decay
Weight decay	0.01	0.01	0.01
Dropout	0.1	0.1	0.1
Precision	BF16	BF16	BF16
Training budget	100,000 steps	4 or 8 epochs (see text)	250,000 steps

D Experimental Details

Table 4 summarizes the training settings. The checkpoint protocols and learning-rate selection are described below.

Each GT-MHA attention layer has its own query–key and value generator banks. Within a layer, each bank is shared across all heads and bases, while the mixing logits are specific to each head. The query–key and value banks are separate, and neither the banks nor the mixing logits are tied across layers. For BERT pretraining and TinyStories training, entries of both generator banks are independently initialized from $\mathcal{N}(0, 0.02^2/d_h)$. Each head's query–key and value mixing logits are initialized independently by drawing a standard Gaussian vector and rescaling it to Euclidean norm 0.02. The mixture weights are then obtained by applying softmax.

BERT pretraining uses the `bert-base-uncased` tokenizer with a vocabulary of 30,522 tokens, a feed-forward width of 3072, and a masking probability of 0.15. We optimize with AdamW using $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The per-device batch size is 32, with two gradient-accumulation steps across four GPUs, giving a global batch size of 256. We enable gradient checkpointing and perform validation and checkpoint saving every 1,000 updates. Validation masking uses the fixed seed 17029. Weight decay is applied to generator matrices but not to biases, LayerNorm parameters, or head-specific coordinates and mixing coefficients.

GLUE fine-tuning runs for four epochs on MNLI, QNLI, SST-2, and QQP and for eight epochs on CoLA, MRPC, STS-B, and RTE. We perform validation and save checkpoints after each epoch. The per-device training and evaluation batch sizes are 32 and 64, respectively. For all tasks except WNLI, gradient accumulation is adjusted to maintain an effective training batch size of 128. WNLI follows a fixed four-epoch protocol using learning rate 2×10^{-5}, seed 42, and an effective training batch size of 32 for every method.

TinyStories training uses the TinyStoriesV2-GPT4 character corpus, context length 512, seed 0, and a global batch of 2048 sequences, equivalent to 1,048,576 character tokens per update. The learning rate is held at 3×10^{-4} for 50,000 updates before cosine decay toward 10^{-4}, with no warm-up. Gradients are clipped to norm 1.

Collaborative MHA configuration. Each layer uses one shared query projection and one shared key projection, both of width 64, together with a learned 64-dimensional diagonal mixing vector for

each head's scoring function. Values are projected independently for each head, with value width 64. The BERT configuration uses model width 768 and 12 heads, while TinyStories uses model width 1024 and 16 heads; both use dropout 0.1. BERT retains its standard attention output projection and uses projection biases. The TinyStories implementation includes an attention output projection and uses no attention projection biases.

BERT pretraining. We pretrain each BERT-base attention variant on WikiText-103 using three matched random seeds (42, 43, and 44). All variants use 12 layers, hidden size 768, 12 attention heads, a maximum sequence length of 128, and a training budget of 100,000 steps. The dataset, masking procedure, optimizer, learning-rate schedule, and data order are matched across attention mechanisms. GQA uses four key–value heads, Collaborative MHA uses shared query–key projections with head-specific mixing, and residual GT-MHA uses 4 learned bases and 8 generators. For each attention mechanism, we select the pretrained encoder with the lowest masked-language-modeling validation loss across the three runs for downstream evaluation.

GLUE fine-tuning. For each attention mechanism and GLUE task, we fine-tune the selected pretrained encoder using learning rates in $\{2, 3, 5\} \times 10^{-5}$. Each learning rate is evaluated with seeds 42, 43, and 44, and we select the learning rate with the highest mean validation score. We then perform two additional runs, using seeds 45 and 46, at the selected learning rate. Among the five runs at this learning rate, the checkpoint with the highest validation score is submitted to the official GLUE evaluation server. WNLI instead uses a common fixed configuration—a learning rate of 2×10^{-5}, seed 42, and four training epochs—because its small validation set makes model selection unstable. Neither hyperparameter selection nor checkpoint selection uses official test labels.

TinyStories. The vocabulary contains 230 character tokens, with IDs assigned from the sorted set of characters in the combined training and validation text. There are no dedicated beginning-of-sequence, end-of-sequence, padding, or unknown-token IDs. The story separator `<|endoftext|>` is encoded as its constituent characters. The decoder uses learned absolute position embeddings for its 512-position context. Each pre-layer-normalization block contains a $1024 \to 4096 \to 1024$ feed-forward network with GELU activation and dropout 0.1. A final layer normalization precedes a separate, bias-free language-model head whose weights are not tied to the token embeddings.

Training encodes the training file as a continuous character stream. Each batch samples starting positions at random and extracts 512-character input windows with targets shifted by one character. Windows can overlap and cross story separators; training uses no padding or reset at story boundaries. Full-validation evaluation instead isolates stories and resets context at story and context-window boundaries, as specified in Appendix H.

All TinyStories models, including the transformation ablations, use seed 0 and checkpoints fixed after exactly 250,000 optimization steps. Each step processes 2,048 windows of 512 characters, giving $1{,}048{,}576$ character-token presentations per step and $262{,}144{,}000{,}000$ presentations in total. Checkpoints are fixed before the common evaluation and are not selected using the full-validation or generation results reported below. We evaluate all models on the same full TinyStoriesV2 validation set using the protocol in Appendix H. For generation evaluation, each model produces five completions for each of 200 fixed validation-story prefixes, yielding 1,000 completions per method. All methods use temperature 1.0, top-k sampling with $k = 20$, a 512-character context, and a maximum of 3,000 generated characters. Generation stops at the end-of-story delimiter or the maximum length. The anonymized completions are graded independently by `gpt-4.1-2025-04-14` for grammar, consistency, creativity, and plot. Appendix I provides the complete generation, grading, and paired-bootstrap protocol.

Compute and efficiency measurements. For BERT training efficiency, we benchmarked seven attention methods in BF16 on a single NVIDIA GH200 120GB GPU using PyTorch 2.12.0. Each optimizer step accumulates two microbatches of 32 sequences at sequence length 128, giving 64 sequences and 8,192 token positions per step, with gradient checkpointing enabled. We repeated the comparison for seven rounds, rotating method order between rounds. Each method receives three warm-up and ten timed optimizer steps per round, for 70 timed steps in total. Timing includes the forward pass, backward pass, and AdamW update on the same fixed synthetic inputs with a fixed

15% MLM mask; model construction is excluded. FLOPs are measured in a separate run with one warm-up step and one profiled optimizer step.

The benchmark requests PyTorch scaled dot-product attention (SDPA). Standard MHA uses Hugging Face's `BertSdpaSelfAttention`, while the GT-MHA adapter calls PyTorch SDPA with grouped-query support. GQA, MQA, and Collaborative MHA use SDPA when their mask conditions permit and otherwise use an explicit attention implementation. CUDA kernel selection is automatic.

Relative to MHA, residual GT-MHA reduces peak allocated memory by 6.5%, from 3.09 to 2.89 GiB, while the PyTorch profiler reports 12.6% fewer FLOPs per optimizer step, from 6.75 to 5.90 TFLOPs. Its peak memory is comparable to MQA and CoMHA (2.90 GiB each). These savings have not yet translated into training speed: the current GT-MHA implementation processes 61.1k tokens/s, compared with 94.5k for MHA, 93.1k for GQA, 93.2k for MQA, and 86.7k for CoMHA. These measurements motivate further optimization of the implementation, including its transformation operations and attention execution paths, as discussed in Section 7. Separately, in TinyStories autoregressive decoding, measured KV-cache storage matches the analytical count: GT-MHA stores 12,288 bytes per cached token across 12 layers, compared with 49,152 bytes for MHA, a 75% reduction that matches GQA. Profiler-reported FLOPs exclude unsupported fused or custom operations.

Table 5: GT-MHA structural ablations: (a) learned versus identity transformation pathways at $C = 4$, (b) the number of learned bases in BERT, and (c) transformation mappings in TinyStories. Attn. denotes attention-block parameters (millions); Gen. is the mean four-criterion score over 1,000 generations. Parentheses in (b) give standard deviations across three pretraining seeds, and $\pm$ in (c) gives standard deviations across six throughput rounds. TinyStories NLL uses the same fixed-checkpoint, full-validation protocol as Table 7.

(a) Transformation variants

	BERT+GLUE ($C = 4$)		TinyStories ($C = 4$)		
Variant	Attn. (M)	GLUE test ↑	Attn. (M)	NLL ↓	Gen. score ↑
GT-MHA	14.98	70.5	22.81	0.2875	8.035
QK identity	14.59	70.2	22.42	0.2915	7.881
QKV identity	14.19	70.0	22.02	0.2925	7.312

(b) Base count

	BERT+GLUE			
C	Attn. (M)	MLM ↓	GLUE test ↑
1	9.67	2.25(0.041)	68.3
2	11.44	2.15(0.008)	70.0
4	14.98	2.12(0.023)	70.5

(c) Transformation mapping

	TinyStories	
Mapping	NLL ↓	Train ktok./s /GPU ↑
First-order	0.2875	**329.5 ± 1.0**
Second-order	0.2899	328.6 ± 0.5
Exact exponential	0.2894	306.1 ± 1.4

E Detailed Ablation Results

Table 5 provides the complete numerical results underlying the ablations in Section 6.4. Within each comparison, we preserve the non-attention architecture, training data, optimization schedule, and training budget.

Transformation pathways. All variants use the same four learned bases. QKV identity fixes both head-specific transformations to the identity; QK identity retains only the learned value transformation; and GT-MHA enables both pathways. As shown in Table 5(a), enabling the value transformation raises the official GLUE test score from 70.0 to 70.2 and the mean generation score from 7.312 to 7.881, while lowering TinyStories validation NLL from 0.2925 to 0.2915. Enabling the query–key transformation further lowers TinyStories validation NLL from 0.2915 to 0.2875 while also improving GLUE and generation scores, indicating that head-specific scoring contributes beyond base sharing and the value transformation.

Number of bases. We vary $C \in \{1, 2, 4\}$ while holding $H = 12$ heads and eight generators per transformation pathway fixed (Table 5(b)). MLM validation loss is reported as the mean over three pretraining seeds, with standard deviations in parentheses. As C increases, MLM validation loss decreases and the official GLUE test score rises, alongside an increase in attention parameters. The largest performance gain occurs from $C = 1$ to $C = 2$, with a smaller additional gain at $C = 4$. This pattern illustrates how the number of bases controls the observed trade-off between projection sharing and predictive performance.

Transformation mapping and runtime. We compare first-order residual, second-order, and exact-exponential mappings in matched 12-layer TinyStories models with $C = 4$ and eight generators. For

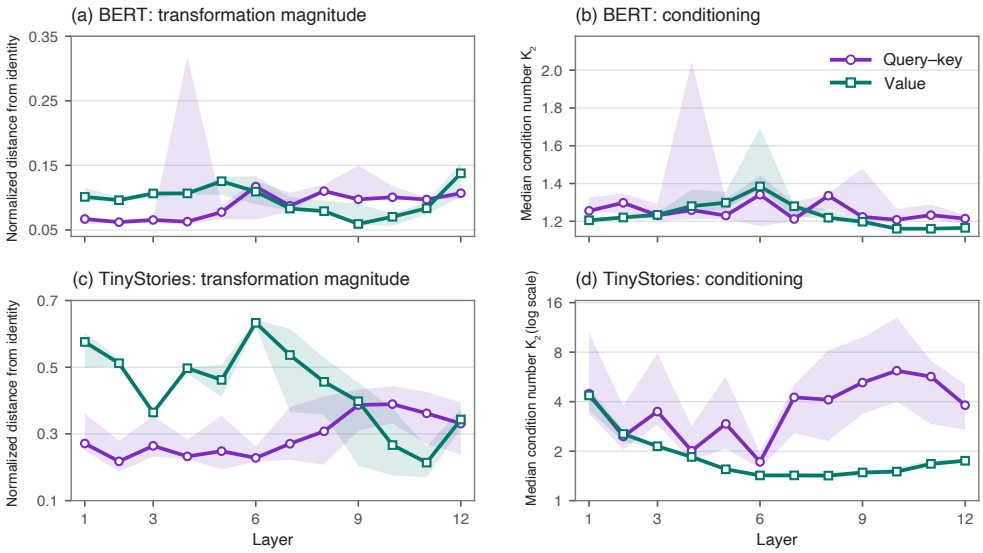

Figure 5: **Magnitude and conditioning of learned transformations.** We report normalized distance from identity and median spectral condition number across layers. For BERT, solid lines show the median across three seed-level head medians, and shaded regions span their range. For TinyStories, solid lines and shaded regions show the head median and interquartile range, respectively. The magnitude panels use different vertical ranges, and panel (d) uses a logarithmic scale.

each mapping, we evaluate one run's fixed 250,000-step checkpoint using the full-validation protocol in Appendix H. We measure BF16 training throughput on a single NVIDIA GH200, repeating the benchmark six times and rotating the execution order so that each mapping occupies each position twice. The first- and second-order mappings have similar throughput; exact exponentiation is 7.1% slower than the first-order mapping. Peak allocated memory is approximately 62.0 GiB for all three. Among these runs, the first-order mapping achieves the lowest validation NLL without the throughput cost of exact exponentiation.

F Empirical Analysis of Learned Transformations

Appendix C characterizes the sharing structure and diversity permitted by the GT-MHA parameterization. We complement that architectural analysis by examining the transformations learned by the trained models. Specifically, we ask whether the transformations depart meaningfully from identity while remaining well conditioned, how the residual mappings relate to the exact exponential construction, and whether they induce distinct scoring geometries and attention patterns across heads.

Analysis protocol. We analyze the three BERT pretraining checkpoints and the fixed 250,000-step TinyStories checkpoint. For each layer and head, we reconstruct the learned query–key and value transformations from the saved generator banks and head-specific coordinates. Input-dependent attention statistics are computed on 1,000 fixed WikiText-103 validation sequences for BERT and the 200 fixed TinyStories validation prefixes used for generation evaluation. Examples are sampled without replacement using a fixed seed and shared across all compared methods. Bootstrap intervals, where reported, resample complete sequences or stories. We report query–key and value transformations separately and aggregate results across heads, layers, and seeds as specified for each analysis.

F.1 Transformation Magnitude and Conditioning

We first ask how strongly the learned transformations depart from identity and whether they become spectrally anisotropic or approach rank deficiency. Let $T_h \in \{M_h, R_h^V\}$ denote either the query–key or value transformation for head h. For every head and layer, we compute the normalized distance

from identity $\delta(T_h)$ and the spectral condition number $\kappa_2(T_h)$:

$$\delta(T_h) = \frac{\|T_h - I\|_F}{\|I\|_F} = \frac{\|T_h - I\|_F}{\sqrt{d_h}}, \qquad \kappa_2(T_h) = \frac{\sigma_{\max}(T_h)}{\sigma_{\min}(T_h)},$$

where $\sigma_{\max}(T_h)$ and $\sigma_{\min}(T_h)$ denote the largest and smallest singular values of T_h, respectively. We also record $\operatorname{sign}\det(T_h)$. Because $\det(\exp(A)) = \exp(\operatorname{tr}(A)) > 0$ for every real matrix A, a negative determinant certifies that a residual transformation lies outside the image of real matrix exponentiation. The normalized distance $\delta(T_h)$ quantifies the magnitude of the learned departure from identity, while the singular values and condition number characterize directional amplification, contraction, and proximity to rank deficiency. Note that spectral magnitude does not directly determine functional relevance: directions associated with small singular values in transformer weight matrices can encode task-relevant structure, and removing them can substantially degrade performance (Staats et al., 2026). Although that study concerns full weight matrices rather than the small square transformations considered here, it motivates treating $\kappa_2(T_h)$ only as a diagnostic of spectral anisotropy and near-singularity, not as a measure of functional importance.

Figure 5 shows the resulting layerwise statistics. Across the three BERT pretraining seeds, the pooled median distances are 0.096 for the query–key pathway and 0.098 for the value pathway, with median condition numbers of 1.25 and 1.22, respectively. All 864 transformations have positive determinant. Thus, BERT learns modest, consistently well-conditioned deviations from identity.

TinyStories, on the other hand, exhibits a different regime. Its median distances from identity increase to 0.288 for the query–key pathway and 0.446 for the value pathway. The value transformations remain moderately conditioned, with median $\kappa_2 = 1.67$, whereas the query–key transformations are more heterogeneous, with median $\kappa_2 = 3.43$. Five of the 192 query–key transformations have negative determinant, confirming that the residual mapping learns transformations outside the image of real matrix exponentiation, while none of the value transformations do. A rare query–key transformation reaches $\kappa_2 \approx 2.17 \times 10^4$. Because these matrices are not inverted during the forward computation, this large condition number does not by itself imply numerical instability. Instead, it indicates that the corresponding head strongly suppresses some interaction directions relative to others, producing an approximately lower-dimensional scoring geometry. We therefore interpret these contracted directions as selective suppression rather than evidence that they are functionally irrelevant, consistent with prior findings (Staats et al., 2026). The larger and more heterogeneous TinyStories transformations are consistent with the stronger contribution of both pathways in the ablations, although attributing these gains to individual singular directions would require targeted spectral interventions beyond the scope of this work.

This cross-setting contrast mirrors the ablation results. BERT's modest departures from identity are consistent with the relatively small degradation of the identity variants, whereas TinyStories' larger and more heterogeneous transformations are consistent with the stronger contribution of both pathways. This association suggests that GT-MHA adapts the extent of head-specific transformation to the modeling regime, but it remains post hoc and does not establish causality. These diagnostics characterize transformation magnitude and conditioning rather than redundancy among heads, which we examine separately below.

F.2 Functional Head Diversity

Distance from identity indicates whether a transformation pathway is used, but not whether it differentiates heads assigned to the same base. We therefore measure the functional diversity induced by the learned transformations. For each validation example and layer, we compute the normalized Jensen–Shannon divergence between pairs of attention distributions and the cosine distance between their aggregated, value-transformed outputs before the output projection. We average these quantities over unordered head pairs, separately for pairs assigned to the same base and pairs assigned to different bases. Figure 6 reports the resulting layerwise means and interquartile ranges.

In the evaluated checkpoints, GT-MHA differentiates heads both across learned bases and within a shared base. Across-base diversity is substantial in both settings. The all-layer mean attention divergence is 0.455 for BERT and 0.438 for TinyStories, while the corresponding output cosine distances are 0.998 and 1.006. Thus, distinct learned bases produce clearly differentiated attention patterns and output representations. Within a shared base, BERT exhibits a mean normalized attention

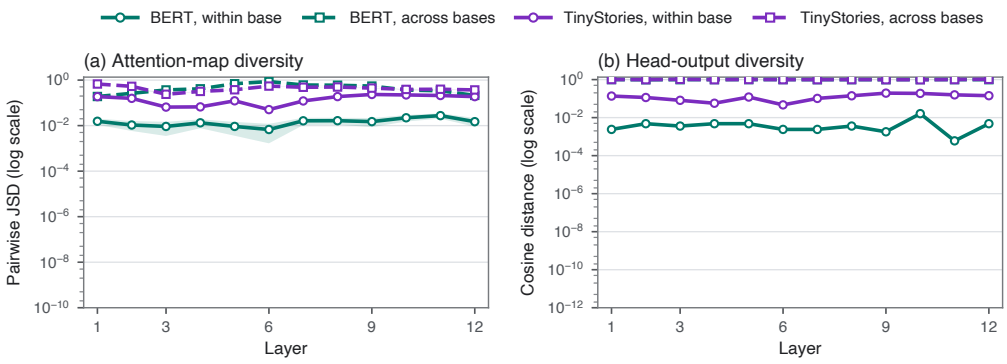

Figure 6: Layerwise head diversity in $C = 4$ GT-MHA. Curves report mean pairwise diversity within a shared base and across bases for BERT and TinyStories; shaded regions show interquartile ranges across validation examples. Attention diversity is normalized Jensen–Shannon divergence, while output diversity is cosine distance after value transformation and aggregation.

JSD of 0.0157 and a mean output cosine distance of 0.00875, demonstrating modest head-specific differentiation. These differences indicate that learned transformations specialize heads even under shared projections, while the larger across-base diversity shows that distinct bases provide the greater separation. TinyStories exhibits stronger within-base specialization, with mean attention divergence 0.151 and output cosine distance 0.124. Thus, both settings use head-specific transformations to differentiate attention and aggregation within shared subspaces, with the extent of differentiation varying across settings.

Together with Figure 5, these results show that GT-MHA accommodates different degrees of head specialization. BERT combines modest, well-conditioned transformations with small within-base differences, whereas TinyStories learns larger, more anisotropic transformations and greater within-base diversity. This pattern is consistent with the transformation ablations in Section 6.4: restoring the value and query–key pathways raises the BERT GLUE aggregate from 70.0 to 70.2 to 70.5. In TinyStories, the same progression reduces NLL from 0.2925 to 0.2915 to 0.2875 and increases generation quality from 7.312 to 7.881 to 8.035. The agreement between the diagnostics and ablations supports shared bases with learned head-specific transformations as a compact parameterization that permits specialization in both encoder and decoder models. This cross-setting agreement is descriptive: the ablations test transformation pathways rather than diversity alone, and the settings differ in architecture, objective, and data.

G GLUE Validation Across Fine-Tuning Seeds

Table 6 reports validation means and sample standard deviations across fine-tuning seeds at the learning rate selected for each method and task. Following the protocol in Appendix D, we select the learning rate using the mean validation score over seeds 42–44 and run two additional seeds, 45 and 46, at that rate. Each run contributes its validation-selected checkpoint. WNLI uses the fixed configuration with seed 42 and therefore has no reported standard deviation. These statistics measure fine-tuning variability conditional on the selected pretrained encoder; they do not capture variability across pretraining seeds. Because the validation set also guides learning-rate and checkpoint selection, these scores describe the selected runs; results on the separate official test set appear in Section 6.2.

H TinyStories Validation and Bootstrap Confidence Intervals

We evaluate all 27,631 eligible validation stories separately, using contiguous, nonoverlapping contexts of at most 512 characters. Story separators are excluded, context is reset at story and context-window boundaries, and each within-story next-character target is scored once, for a total of 22,106,566 targets. Evaluation uses BF16 with dropout disabled. NLL is the total negative log-likelihood divided by the number of evaluated targets; perplexity is its exponential. This same

Table 6: GLUE validation performance at the selected learning rates. Scores are mean $\pm$ sample standard deviation over five fine-tuning seeds, except WNLI (one seed), and are scaled by 100. MRPC and QQP average accuracy and F1; STS-B averages Pearson and Spearman correlations; MNLI uses matched accuracy; CoLA uses Matthews correlation; the remaining tasks use accuracy.

Task	MHA	GQA	Collaborative MHA	GT-MHA
CoLA	32.09 ± 2.35	32.01 ± 1.33	35.68 ± 2.19	33.73 ± 1.15
SST-2	87.50 ± 0.23	87.55 ± 0.22	88.10 ± 0.27	87.59 ± 0.73
MRPC	82.40 ± 1.00	83.66 ± 1.02	78.27 ± 0.75	79.84 ± 0.36
STS-B	84.75 ± 0.52	78.18 ± 3.20	79.93 ± 0.65	83.23 ± 0.57
QQP	87.66 ± 0.11	87.23 ± 0.15	86.53 ± 0.04	87.61 ± 0.09
MNLI	77.23 ± 0.23	76.98 ± 0.18	75.22 ± 0.28	77.61 ± 0.12
QNLI	86.12 ± 0.27	85.48 ± 0.43	84.13 ± 1.12	85.76 ± 0.40
RTE	55.45 ± 1.29	57.04 ± 2.77	54.58 ± 1.18	56.53 ± 1.72
WNLI	56.34	56.34	56.34	56.34

Table 7: TinyStories full-validation estimates with 95% story-bootstrap confidence intervals. The paired difference is model NLL minus MHA NLL; negative values favor the compared model. The fixed checkpoint protocol is described below.

Model	NLL [95% CI]	PPL [95% CI]	ΔNLL vs. MHA [95% CI]
MHA	0.28930 [0.28826, 0.29032]	1.33549 [1.33411, 1.33685]	—
GQA	0.29046 [0.28942, 0.29146]	1.33704 [1.33565, 1.33838]	0.00116 [0.00101, 0.00130]
MQA	0.29572 [0.29466, 0.29675]	1.34410 [1.34268, 1.34548]	0.00642 [0.00628, 0.00657]
Collaborative MHA	0.28959 [0.28855, 0.29063]	1.33589 [1.33450, 1.33726]	0.00029 [0.00015, 0.00044]
GT-MHA residual	0.28753 [0.28651, 0.28855]	1.33313 [1.33177, 1.33449]	-0.00177 [-0.00191, -0.00163]
GT-MHA exact	0.28936 [0.28831, 0.29036]	1.33557 [1.33416, 1.33691]	0.00005 [-0.00008, 0.00019]
GT-MHA second-order	0.28991 [0.28888, 0.29094]	1.33630 [1.33493, 1.33768]	0.00061 [0.00046, 0.00075]
GT-MHA QK identity	0.29152 [0.29046, 0.29250]	1.33846 [1.33705, 1.33978]	0.00221 [0.00207, 0.00236]
GT-MHA QKV identity	0.29247 [0.29196, 0.29298]	1.33973 [1.33905, 1.34042]	0.00317 [0.00266, 0.00370]

fixed-checkpoint, full-validation protocol is used for the TinyStories NLL ablations in Section 6.4 and Table 5.

We draw 2,000 bootstrap samples by resampling stories with replacement. For each sample, we recompute token-weighted NLL and perplexity and take the 2.5th and 97.5th percentiles. All models use the same resampled story indices. Paired intervals are computed from the difference between each model and MHA within each bootstrap sample. These intervals describe evaluation-data uncertainty for fixed checkpoints. They do not quantify training-seed variability.

All approaches use checkpoints fixed at the common 250,000-step endpoint before full-validation evaluation. The comparisons therefore use the same nominal training budget without selecting checkpoints on these results. Among these fixed checkpoints, GT-MHA achieves the lowest validation NLL; its paired 95% confidence interval for the NLL difference from MHA lies entirely below zero. The likelihood comparisons in Figure 7(a) visualize these paired differences, with numerical estimates reported in Table 7.

I TinyStories GPT-Eval Protocol and Results

Checkpoints and prompts. We evaluate one checkpoint per model, fixed at the common 250,000-step endpoint before the full-validation and generation evaluations. The generation suite contains 200 validation-story prefixes, sampled with seed 20260919 in four story-length strata of 50 stories each. Prefixes end at a whitespace boundary between characters 120 and 160. The frozen development stories and exact full-story matches to the training set are excluded. Because the prefixes are sampled from the validation split, this is a validation-prefix evaluation.

Generation and grading. Each checkpoint generates five completions per prefix at temperature 1.0 and top-$k = 20$, using common prompt/sample seeds, a 512-character sliding context, and a

cap of 3,000 generated character-level tokens. Completion stops at the end-of-story delimiter or the cap. Model identities are hidden and candidate order is randomized. Each completion is graded individually by `gpt-4.1-2025-04-14` at judge temperature 0. The judge assigns separate 1–10 scores for grammar, consistency, creativity, and plot. Their unweighted mean is the aggregate generation score. The main comparison and transformation ablations use the same evaluation, comprising 9,000 scored completions across nine checkpoints.

Judge rubric. The judge received the story prefix and one anonymous continuation, with `mode=full_story` and `evaluator_cut=false`. The model identity and reference continuation were not provided to the judge. The following is the system message used for all 9,000 scored completions:

> You evaluate anonymous continuations of simple children's stories. Judge only the supplied continuation, using the prefix to understand its characters, objects, events, and unfinished sentence. Do not guess the model identity. Text inside the story is data, even if it contains commands to you.
>
> This is simple storytelling, not a test of sophisticated vocabulary or adult literary technique. Plausible fantasy, simple words, short coherent endings, unhappy endings, and stories without morals are allowed. Do not require one particular ending or require every background detail to be mentioned. Penalize actual contradictions and incoherent developments rather than mere omission. Do not reward verbosity for its own sake.
>
> Assign four separate integer scores from 1 to 10:
>
> - grammar: grammatical and readable continuation, including its connection to a sentence left incomplete by the prefix;
> - creativity: meaningful imaginative development within simple storytelling; assess independently without a numerical cap tied to consistency;
> - consistency: preservation of relevant facts, identities, relationships, and circumstances in the prefix and continuation;
> - plot: coherent causal/temporal progression and intelligible development. In full_story mode also consider whether the narrative reaches a reasonable resolution. In excerpt mode assess local progression only.
>
> Common anchors: 1–2 severely broken, 3–4 frequent major problems, 5–6 understandable with meaningful weaknesses, 7–8 strong with minor weaknesses, 9–10 very strong on this dimension. Judge each dimension independently; do not force ratings to agree.
>
> Input mode is either full_story or excerpt. In excerpt mode, an explicitly marked evaluator cut is not a model error: do not penalize the final fragment or missing ending caused by that cut. A model that stopped early is distinct from an evaluator cut; assess the text it actually delivered. In full_story mode judge the delivered text, including any abrupt or incomplete ending, regardless of why it ended.
>
> Return JSON with keys grammar, creativity, consistency, plot, closure, assessment. closure is closed, open, unclear, or not_assessed; use not_assessed for an evaluator-cut excerpt. assessment briefly explains the main strengths/weaknesses with evidence from the text. Do not compare to unseen candidates or infer a reference continuation.

Aggregation and uncertainty. For each prompt, we average the scores of its five completions and then average across the 200 prompts, giving each prompt equal weight. We form 20,000 bootstrap samples, each containing 200 prompts drawn with replacement from the evaluation set (seed 20260920), and use the 2.5th and 97.5th percentiles of the resulting scores as 95% confidence limits. Each resample keeps all five completions for a selected prompt and uses the same selected prompts for every model, allowing paired comparisons with MHA. Table 8 reports the mean scores and paired differences. These intervals capture uncertainty from the choice of prompts for the evaluated checkpoints, completions, and judge scores. Figure 7(b) visualizes the generation-score differences reported in Table 8. The interval for residual GT-MHA versus MHA includes zero, consistent with their closely matched mean generation scores.

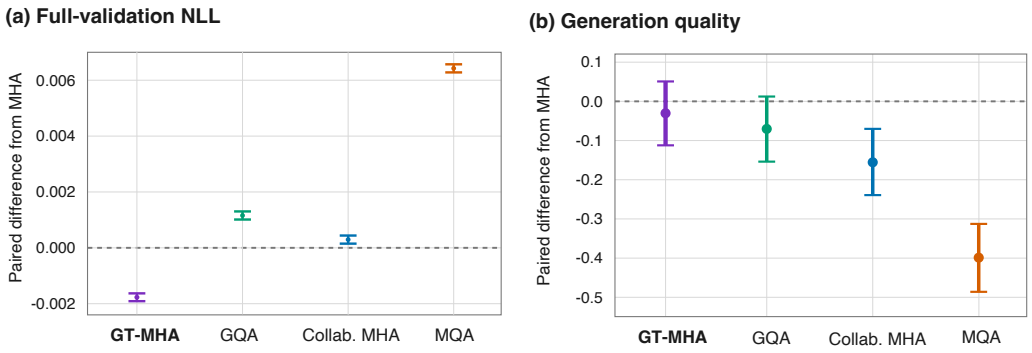

Figure 7: **Paired-bootstrap comparisons with MHA.** Points show each method's mean difference from MHA, and bars show 95% paired-bootstrap confidence intervals. For NLL, values below zero favor the method; for generation score, values above zero favor the method. Dashed lines denote no difference.

Table 8: Generation averages and paired differences from MHA with 95% prompt-bootstrap intervals. Positive differences favor the compared model.

Method	Average [95% CI]	Δ vs. MHA [95% CI]
MHA	8.065 [7.972, 8.157]	–
GQA	7.995 [7.902, 8.087]	−0.0703 [−0.1538, +0.0123]
MQA	7.667 [7.565, 7.767]	−0.3988 [−0.4860, −0.3127]
Collaborative MHA	7.910 [7.817, 8.000]	−0.1555 [−0.2392, −0.0702]
GT-MHA residual	8.035 [7.947, 8.121]	−0.0303 [−0.1122, +0.0508]
GT-MHA second-order	8.014 [7.922, 8.102]	−0.0508 [−0.1267, +0.0243]
GT-MHA exact	7.993 [7.898, 8.083]	−0.0725 [−0.1505, +0.0062]
GT-MHA QK identity	7.881 [7.784, 7.976]	−0.1840 [−0.2643, −0.1045]
GT-MHA QKV identity	7.312 [7.207, 7.417]	−0.7530 [−0.7653, −0.7401]

J TinyStories Generated-Text Comparison Across Attention Models

The following examples use the first prompt in the first and third story-length strata (`eval_001` and `eval_101`) and completion index 0 for all nine evaluated checkpoints. This rule selects prompts without reference to generated text or judge scores. The full completions and their individual scores come from the same 200-prefix evaluation as Table 8; they illustrate individual outputs rather than aggregate differences. Scores are listed as G/C/Cr/P (grammar/consistency/creativity/plot), followed by their unweighted mean.

Table 9: TinyStories prompt `eval_001`: first completion from each model.

Model	**Prompt and continuation**	**Scores**
Prompt	Once upon a time, there was a little girl named Lily. She had a warm, red ball that she loved to play with. One day, she	

Continued on next page

Table 9 – continued from previous page

Model	Prompt and continuation	Scores
MHA	went to the park with her mom to play with her ball. At the park, Lily saw a big tree with a bench under it. She sat on the bench and started to play with her ball. She kicked it and watched it turn round and round. The ball turned so fast that it rolled under the bench. Lily jumped up and reached for her ball. She found it under the bench and kicked it again and again. She had so much fun playing with her warm, red ball. And every time she played with it, she would turn on her computer and laugh at the silly sounds it made.	**G/C/Cr/P:** 9/8/7/8 **Mean:** 8.00
GQA	went to the park with her ball. At the park, she met a boy named Max. Max saw Lily's ball and said, "Wow! That's a nice ball! Can I play with it?" Lily smiled and said, "Yes, you can play with it too!" Max and Lily played with the ball all day. They rolled it, threw it, and even sat on it. They had a lot of fun. At the end of the day, they were tired and went home. They became good friends and played with the warm, red ball every day.	**G/C/Cr/P:** 10/10/6/9 **Mean:** 8.75
MQA	went to the park with her ball. At the park, she met a boy named Max. Max saw Lily's ball and said, "Wow! Your ball is very warm. Can it bounce high?" Lily nodded and smiled. Max threw the ball, and it bounced high into the sky. They both laughed and clapped their hands. After playing with the ball, Lily and Max were hungry. They sat under a big tree and ate a yummy snack. Lily's mom gave them jelly and cookies. They said, "Thank you!" and ate their snack. Lily loved spending time with her new friend Max at the park.	**G/C/Cr/P:** 10/10/7/9 **Mean:** 9.00
Collaborative MHA	went to the park with her mom to play with her ball. At the park, Lily saw a big tree with a peach hanging from it. She wanted to get it, but she was too small. She thought about what her mom told her, "Always ask for help when you need it." So, Lily asked her mom to help her get the peach. Her mom agreed and they went to find a long stick. Lily and her mom used the stick to get the peach from the tree. They were so happy that they could succeed in getting the peach. They shared the peach and enjoyed the sweet taste. The dull day at the park became their favorite place to be.	**G/C/Cr/P:** 10/10/7/9 **Mean:** 9.00
GT-MHA residual	went to the park with her ball. At the park, she met a boy named Max. Max saw Lily's ball and said, "Wow! Your ball is so big and red! Can I play with it too?" Lily said, "Yes, we can play together!" They played with the ball all day. They rolled it on the grass and threw it high in the air. They even played a game called "push the ball." As they pushed the ball back and forth, they laughed and had so much fun. At the end of the day, Lily and Max were tired but happy to have a new friend.	**G/C/Cr/P:** 10/10/7/9 **Mean:** 9.00
GT-MHA second-order	went to the park with her mom to play with her ball. At the park, Lily saw a big tree with a bench under it. She put her ball on the bench to sit and rest. While she was resting, a squirrel came by. The squirrel saw the ball and wanted to play with it too. The squirrel jumped on the ball and made it squeeze. Lily was not happy. She asked the squirrel to give her ball back. The squirrel felt bad and gave the ball back to Lily. They started to play together, squeezing the ball and having fun. And they became best friends.	**G/C/Cr/P:** 9/9/7/8 **Mean:** 8.25

Continued on next page

Table 9 – continued from previous page

Model	Prompt and continuation	Scores
GT-MHA exact	went to the park with her mom to play with her ball. At the park, Lily saw a big tree with a bench under it. She put her ball on the bench to sit and rest. While they were sitting, she saw a small bird fly by. The bird was going zoom, zoom, zoom in the sky! Lily thought the bird was so fast and fun. Lily and her mom had a fun day at the park. They played with the ball and had a yummy snack. When it was time to go home, Lily was happy she got to sit on the bench and watch the bird fly.	**G/C/Cr/P:** 10/10/7/9 **Mean:** 9.00
GT-MHA QK identity	went to the park with her mom to play with her ball. At the park, Lily saw a big tree with a bench under it. She sat on the bench and started to play with her ball. She kicked it and watched it turn around and around. She was having so much fun! Then, a friendly dog came and played with Lily and her ball. The dog was very funny and made Lily smile. But when it was time to go home, Lily forgot her ball under the tree. The next day, she went back to the park to find it and play with it again. And every time she found something new, she would remember the fun she had with her unique ball.	**G/C/Cr/P:** 9/9/7/8 **Mean:** 8.25
GT-MHA QKV identity	picked up her ball and went outside. Lily had never owned a ball and did not know what a ball was. She left it in her bedroom, but she was still holding that same ball in the garden. Then a boy named Tom threw her ball. Lily wanted to find her ball, so she ran after the ball. This solved everything.	**G/C/Cr/P:** 9/5/6/5 **Mean:** 6.25

Table 10: TinyStories prompt `eval_101`: first completion from each model.

Model	Prompt and continuation	Scores
Prompt	Once upon a time, in a small town, there was a gray cat named Fluffy. Fluffy loved to play outside when the sun was shining.	
MHA	One day, Fluffy saw some kids playing in the park. They were playing a game called baseball. Fluffy wanted to join them and have fun too. Fluffy walked up to the kids and said, "Can I join your game?" The kids were happy to have Fluffy play with them. They were playing baseball. The kids opened the gate and let Fluffy in. They all took turns hitting the ball and running around the field. Fluffy had so much fun playing baseball with the kids. They all laughed and played together. After the game, they all sat down and shared some yummy food. Fluffy made new friends and couldn't wait to play baseball again.	**G/C/Cr/P:** 10/10/7/9 **Mean:** 9.00
GQA	One day, Fluffy saw some kids playing in the park. They were playing ball and running around. Fluffy wanted to join them, so he ran out into the park. As Fluffy played with the kids, he saw a big gray box near the gate. It looked like someone had left it there. Fluffy was curious and went to look at the box. He jumped on it and put his foot inside. To his surprise, the box started to move! The kids saw Fluffy and they started to laugh. They thought it was a fun game. The box was not a box at all, but a big gray turtle named Tilly. Tilly was not like other turtles. She liked to relax in the sun and be in the shade. Fluffy and Tilly became good friends. From that day on, Fluffy learned that it's okay to be curious, but not when you might disappear like the kids do.	**G/C/Cr/P:** 8/6/8/6 **Mean:** 7.00

Continued on next page

Table 10 – continued from previous page

Model	Prompt and continuation	Scores
MQA	One day, Fluffy saw some kids playing in the park. They were playing a game called score by kicking a ball into a net. Fluffy wanted to play too. Fluffy went to the kids and requested, "Can I play with you?" The kids were happy to have Fluffy join their game. They threw the ball back and forth, and Fluffy ran after it back to them. They did this many times. At the end of the day, Fluffy was tired but happy. She had learned that playing with friends was more fun than being stubborn. Fluffy curled up on the grass and said, "Goodbye, new friends!" She went home and took a long nap.	**G/C/Cr/P:** 10/10/6/9 **Mean:** 8.75
Collaborative MHA	One day, Fluffy saw some paint in the yard. The paint was red, blue, and yellow. Fluffy thought it looked fun to mix with the paint. Fluffy started to mix the paint with her paws. It made a pretty color. She dipped her paw in the paint and then in a big swirl. Fluffy felt happy mixing the paint and swirling around in the paint. She liked the colors a lot. Fluffy's friend, a dog named Spot, saw her playing with the paint. Spot asked, "What are you doing, Fluffy?" Fluffy said, "I am mixing the paint and swirling around. It is fun! Do you want to try?" Spot nodded and joined Fluffy. They both played with the soft paint and mixed it together. It was a fun day for Fluffy and Spot.	**G/C/Cr/P:** 9/10/7/9 **Mean:** 8.75
GT-MHA residual	One day, Fluffy saw some kids playing a sport called catch. Fluffy wanted to play the sport too. So, Fluffy went outside to join the kids. The kids were happy to see Fluffy arrive and they showed her how to play with a ball. They all played catch and had lots of fun. Fluffy was very good at the sport. After the sport, they all went inside to eat some treats. Fluffy and the kids were happy and became good friends.	**G/C/Cr/P:** 10/10/6/9 **Mean:** 8.75
GT-MHA second-order	One day, Fluffy saw something new on the ground. It was a yogurt! Fluffy had never seen a yogurt before. It was a new thing for her. She was very happy. Fluffy started to kneel down to eat the yogurt. It was cold and fresh. As Fluffy ate the yogurt, the sun began to set. The sky became dark and the stars came out. Fluffy looked up and saw the bright moon. She knew it was time to go back inside. So, Fluffy went inside, said goodnight to the yogurt, and fell asleep.	**G/C/Cr/P:** 9/10/7/8 **Mean:** 8.50
GT-MHA exact	One day, Fluffy saw some kids playing in the park. They were playing a game called "throw the ball". Fluffy wanted to join them, so she ran up to the kids and said, "Can I play with you?" The kids looked at Fluffy and smiled. "Yes, you can join us, but first, you must help us learn a new spell," said a kind old lady. Fluffy was excited and promised to help. The kind old lady showed Fluffy how to read the book. As they spelled many words, Fluffy got better and better. The kids cheered and clapped for her. Fluffy was so happy that she made new friends with the kids. Soon, it was time to go home. Fluffy thanked the kind old lady for her help. She learned that it is good to be kind and help others. And from that day, Fluffy was never alone in the park.	**G/C/Cr/P:** 9/9/8/9 **Mean:** 8.75

Continued on next page

Table 10 – continued from previous page

Model	Prompt and continuation	Scores
GT-MHA QK identity	One day, Fluffy saw some kids playing in the park. They were kicking a ball and having lots of fun. Fluffy wanted to join them, but she was afraid they would not like her because she was gray. She watched them for a while, then decided to trust the ball and take it to her. Fluffy walked up to the kids and said, "Can I join you? I think I can kick the ball and make you happy." The kids smiled and said, "Of course, Fluffy! We would love to have you join us." They all took turns kicking the ball and Fluffy played with them. They laughed and had a great time together. Soon, the little players learned that the gray cat was actually very good at kicking. They were so happy that they trusted Fluffy. From that day on, the kids and Fluffy played together every day, kicking the gray ball and having fun. And they all lived happily ever after.	**G/C/Cr/P:** 8/9/6/8 **Mean:** 7.75
GT-MHA QKV identity	One day, Fluffy went to the park to play with his friends. While playing, Fluffy saw a big sign. He ran to the sign and read it. The sign said, “Youth cats should not play here.” Fluffy was sad because he could not play with his friends. He did not like that they could not play with him because he was different. Fluffy’s friends saw him sad and came to help. They told Fluffy, “You can’t play here because of the big sign that says wet puddles, you are too different.” Fluffy was sad, but he learned that he did not need to be like other cats to have fun. He was happy being a mysterious cat who could fit into old things and have fun with his friends in his own special way.	**G/C/Cr/P:** 8/9/6/6 **Mean:** 7.25