WISCA: A Lightweight Model Transition Method to Improve LLM Training via Weight Scaling

Jiacheng Li[1], **Jianchao Tan**[1] *, **Zhidong Yang**[3], **Pingwei Sun**[1], **Feiye Huo**[1], **Jiayu Qin**[1]
Xiangyu Zhang[2], **Maoxin He**[4], **Guangming Tan**[2], **Weile Jia**[2], **Xunliang Cai**[1], **Tong Zhao**[2] *,
[1]Meituan, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Hong Kong University of Science and Technology, Hong Kong SAR, China
[4]Xiamen University, Xiamen, China
{lijiacheng14, tanjianchao02}@meituan.com
{jiaweile, tongzhao.zhaot}@gmail.com

Abstract

Transformer architecture gradually dominates the LLM field. Recent advances in training optimization for Transformer-based large language models (LLMs) primarily focus on architectural modifications or optimizer adjustments. However, these approaches lack systematic optimization of weight patterns during training. Weight pattern refers to the distribution and relative magnitudes of weight parameters in a neural network. To address this issue, we propose a Weight Scaling method called WISCA to enhance training efficiency and model quality by strategically improving neural network weight patterns—without changing network structures. By rescaling weights while preserving model outputs, WISCA indirectly optimizes the model's training trajectory. Experiments demonstrate that WISCA significantly improves convergence quality (measured by generalization capability and loss reduction), particularly in LLMs with Grouped Query Attention (GQA) architectures and LoRA fine-tuning tasks. Empirical results show **5.6%** average improvement on zero-shot validation tasks and **2.12%** average reduction in training perplexity across multiple architectures.

1 Introduction

The weight pattern is defined as the arrangement and scaling of parameters across layers in a neural network, which plays a pivotal role in the convergence of neural networks. For example, although a two-layer fully connected network theoretically can approximate any function (universal approximation theorem) Barron (1993), its performance on complex tasks is often suboptimal due to poorly structured weight patterns. In contrast, modern architectures (e.g., CNNs, GNNs, RNNs, Transformers) achieve superior results by implicitly optimizing weight patterns for specific tasks. These architectures reduce optimization complexity and enhance representational capability through structured designs (e.g., local receptive fields in CNNs LeCun et al. (2002), attention mechanisms in Transformers Vaswani et al. (2017)), demonstrating that an effective weight pattern is a key factor in their success.

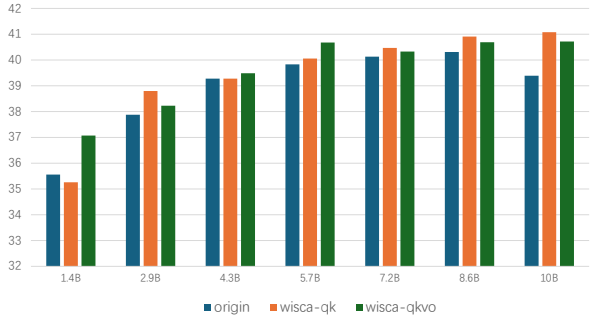

Figure 1: The comparisons of average zero-shot evaluation results at different training steps using WISCA and original methods on llama-moe-5B-A0.8B. All metrics can be seen in Appendix.

A typical example of a poor weight pattern in a neural network that leads to suboptimal model performance is the sharp minimum problem. As demonstrated in Keskar et al. (2016), training with excessively large batch sizes increases the likelihood of converging to sharp minima. Models trapped in sharp minima exhibit significantly worse generalization on test datasets compared to those in flat minima. This discrepancy arises from the bias between the training and test sets: Firstly, the loss landscapes computed on training and test datasets diverge significantly due to their distributional differences. Secondly, because of the distributional relationship between the two datasets being close, the losses of them exhibit relatively similar but inconsistent trends in parameter spaces. The loss landscape curves of them are parallel but non-overlapping trajectories. Thus, the side impact of sharp minima on generalization is apparent. It amplifies sensitivity to outliers in the dataset and weakens the model's robustness as a result.

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 1589–1601
July 2-7, 2026

In non-convex optimization, rugged convergence paths, characterized by sharp minima and highly curved regions in the loss landscape, pose significant challenges to gradient-based optimization. Such paths force the optimizer to wander through narrow valleys and saddle points, leading to premature convergence to suboptimal solutions with poor generalization (e.g., sharp minima). In contrast, a smooth, flat convergence path allows the gradients to guide parameters in approaching a wider minimum, where small perturbations in the weights have minimal side-impact on the loss. This stability inherently improves generalization, as models in flat regions are less sensitive to outliers in the dataset. This difference highlights the critical role of loss landscape geometry in successfully converging to a wider minimum.

From the perspective of optimizing weight patterns to improve neural network training trajectories in the loss landscape and the results of convergence, we propose the Equivalent Model Theory and an optimization strategy for weight patterns (WISCA). Equivalent Model Theory gives the definition of equivalent models. Consider two models; if they have the same architecture and derive the same output with the same input despite the differences in weight configurations of them. Such two models can be regarded as equivalent models. During training, models can transition to equivalent models, and the training process will continue. WISCA introduces a systematic strategy to guide such transitions to superior weight patterns, thereby indirectly reshaping the training process dynamically. By prioritizing weight configurations that align with smoother optimization landscapes, WISCA enhances the likelihood of converging to a flat minimum with superior generalization performance.

Our contributions can be summarized as follows:

- We propose the Equivalent Model Theory, which establishes a theoretical foundation for enhancing model training performance through weight pattern optimization.
- Based on the proposed theory, we propose a novel model transition strategy called WISCA, which enables Transformer-based models and LoRA architectures to dynamically adjust their weight patterns to more optimal configurations during training while remaining equivalent to the unadjusted model.
- Extensive experimental results show that our WISCA can improve model performance through dynamically adjusting weights by finding equivalent models.

2 Theory

2.1 Sharp Minima leads to Worse Generalization

Theorem 1 (Generalization Gap of Sharp vs. Flat Minima)**.** *Let θ_1 and θ_2 be two models with identical training loss $L_{train}(\theta_1) = L_{train}(\theta_2) = L_0$. If θ_1 resides in a sharp minimum (defined as $\exists\, \mathcal{N}(\epsilon)$, for $\forall \epsilon \in \mathcal{N}(\epsilon)$, where $\|\epsilon\| \to 0$, $L(\theta_1 + \epsilon) - L(\theta_1) > L(\theta_2 + \epsilon) - L(\theta_2)$), then the expected validation loss satisfies:*

$$\mathbb{E}[L_{val}(\theta_1)] > \mathbb{E}[L_{val}(\theta_2)] \tag{1}$$

Proof. The proof proceeds in five steps:

Taylor Expansion of Loss Landscape: For a small perturbation $\epsilon \in \mathbb{R}^d$, the loss at $\theta + \epsilon$ is approximated by:

$$L(\theta + \epsilon) \approx L(\theta) + \frac{1}{2}\epsilon^\top H(\theta)\epsilon \tag{2}$$

where $H(\theta) = \nabla^2 L(\theta)$ is the Hessian matrix. At minima ($\nabla L(\theta) = 0$), the first-order term vanishes.

Sharpness Characterization: The sharpness condition implies that for all ϵ:

$$\epsilon^\top H(\theta_1)\epsilon > \epsilon^\top H(\theta_2)\epsilon \quad \Rightarrow \quad H(\theta_1) \succeq H(\theta_2) \tag{3}$$

where $\succeq$ denotes the positive semi-definite ordering.

Modeling Validation Loss: Assume validation data introduces a Gaussian perturbation $\delta \sim \mathcal{N}(0, \sigma^2 I)$ to the parameters. The validation loss is approximated as:

$$L_{\text{val}}(\theta) \approx L_0 + \frac{1}{2}\delta^\top H(\theta)\delta \tag{4}$$

Expectation Calculation: Taking expectation over δ:

$$\mathbb{E}[L_{\text{val}}(\theta)] = L_0 + \frac{\sigma^2}{2}\text{Tr}(H(\theta)) \tag{5}$$

where $\text{Tr}(H(\theta))$ is the trace of the Hessian.

Inequality Derivation: Since $H(\theta_1) \succeq H(\theta_2)$, it follows that $\text{Tr}(H(\theta_1)) \geq \text{Tr}(H(\theta_2))$. Thus:

$$\begin{aligned} &\mathbb{E}[L_{\text{val}}(\theta_1)] - \mathbb{E}[L_{\text{val}}(\theta_2)] \\ &= \frac{\sigma^2}{2}\left(\text{Tr}(H(\theta_1)) - \text{Tr}(H(\theta_2))\right) > 0 \end{aligned} \tag{6}$$

□

2.2 Equivalent Model

Definition 1 (Equivalent Models)**.** *Let $\mathcal{F}$ denote a neural network architecture subject to parameter space Θ, input space $\mathcal{X}$, and output space $\mathcal{Y}$. Two parameter configurations $\theta_1, \theta_2 \in \Theta$ are called* ***equivalent models*** *if they satisfy:*

1. ***Architectural Consistency****: θ_1 and θ_2 belong to the same architecture $\mathcal{F}$.*

2. ***Functional Equivalence****: For all inputs $\mathbf{x} \in \mathcal{X}$,*
$$F(\mathbf{x}; \theta_1) = F(\mathbf{x}; \theta_2) \tag{7}$$
where $F(\cdot; \theta)$ denotes the function of forward propagation for architecture $\mathcal{F}$ with parameters θ.

3. ***Parameter Distinction****: $\theta_1 \neq \theta_2$ (i.e., their weight patterns differ).*

Equivalent models represent distinct points in the parameter space Θ that map to the same functional behavior in the output space $\mathcal{Y}$. This concept enables optimization strategies (e.g., our proposed WISCA) to transition between equivalent models during training, effectively reshaping weight patterns without altering functional outputs.

For example, consider a two-layer fully connected network where:

1. Layer l uses ReLU$(\cdot)$ activation with zero bias,

2. Parameters are $(\mathbf{w}^{(l)}, \mathbf{w}^{(l+1)})$ and $(\alpha\mathbf{w}^{(l)}, \alpha^{-1}\mathbf{w}^{(l+1)})$ for $\alpha > 0$.

For any input $\mathbf{x}$, the outputs are:

$$\theta_1 : \quad \mathbf{w}^{(l+1)}\text{ReLU}\left(\mathbf{w}^{(l)}\mathbf{x}\right) \tag{8}$$

$$\theta_2 : \quad \alpha^{-1}\mathbf{w}^{(l+1)}\text{ReLU}\left(\alpha\mathbf{w}^{(l)}\mathbf{x}\right) \tag{9}$$

Since ReLU$(\alpha\mathbf{z}) = \alpha$ReLU$(\mathbf{z})$ for $\alpha > 0$, θ_2 simplifies to:

$$\alpha^{-1}\mathbf{w}^{(l+1)} \cdot \alpha\text{ReLU}\left(\mathbf{w}^{(l)}\mathbf{x}\right) = \mathbf{w}^{(l+1)}\text{ReLU}\left(\mathbf{w}^{(l)}\mathbf{x}\right)$$

which matches θ_1 exactly.

Thus, θ_1 and θ_2 are **equivalent models** (Definition 1):

- Identical architecture and outputs for all inputs
- Distinct weight patterns ($\mathbf{w}^{(l)} \neq \alpha\mathbf{w}^{(l)}$)

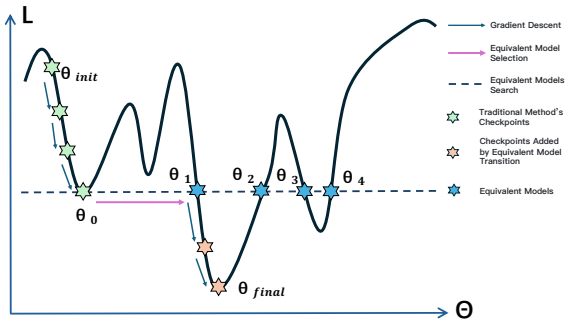

Figure 2: **Optimization via Equivalent Model Transition.** In the ideal case, the global optimum θ_{final} can be achieved through this strategy. Key notations: θ_{init} (initialized model), θ_0 (final checkpoint under conventional optimization), $\theta_1, \ldots, \theta_n$ (all equivalent models explored globally), θ_{final} (global optimum). After identifying equivalent models, the optimal candidate (e.g., θ_1) is selected for continued training based on convergence potential. Theoretically, convergence to θ_{final} is guaranteed if all equivalent models are evaluable.

An ideal implementation of the equivalent model strategy for optimizing neural network training trajectories consists of the following steps:

Escape from Local Minima: When a first-order optimizer (e.g., SGD, Adam) becomes trapped in a sharp local minimum θ_0 with poor generalization, systematically explore the set of equivalent models $[\theta_1, \theta_2, ..., \theta_n]$ (as defined in Definition 1).

Model Selection: From the candidate set $[\theta_1, \theta_2, ..., \theta_n]$, select a model θ_k that lies in a "flatter" region of the loss landscape, since such regions are empirically linked to better generalization.

Training Transition: Resume training from θ_k instead of θ_0, leveraging the functional equivalence of the models to maintain task performance while improving optimization dynamics.

Theoretically, if one can identify all equivalent model strategies for a neural network architecture, the model can converge to the global optimum. However, it is quite challenging to implement in practice. We propose a strategy named WISCA, model transition during training that identifies partial equivalent models, which have demonstrated outstanding performance in extensive experiments.

3 WISCA

For an input sequence $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$ with n tokens and d_{model}-dimensional embeddings, the self-attention operation is formulated as follows:

$$(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{X}\,(\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v) \tag{10}$$

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \quad (11)$$

$$\text{Output} = \text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})\mathbf{W}_o \quad (12)$$

For the θ-component ($\theta = \{W_q, W_k\}$), we model its loss landscape as:

$$\mathcal{L}(\mathbf{Q}, \mathbf{K}) = (\mathbf{Q}\mathbf{K} - C)^2, \quad C \in \mathbb{R}^+. \quad (13)$$

where, C represents the ideal value of $\mathbf{QK}$ (e.g., $C = 1$ for normalized attention).

Under the assumption of perfect equivalent model strategies, parameters can traverse *any point* on the contour lines of $\mathcal{L}(\mathbf{Q}, \mathbf{K}) = \text{const}$ (Figure 3). On the contour lines of the loss landscape, the region where $\mathbf{Q} = \mathbf{K}$ exhibits the flattest geometry. This is reflected in the top-down contour lines, where the distance between adjacent contour lines reaches the maximum when $\mathbf{Q} = \mathbf{K}$.

We assume that flatter regions of the loss landscape are of higher quality than sharper, more rugged regions at the same loss level. This superiority manifests not only in generalization performance but also in the efficiency of continued training. The core idea of WISCA is to adjust the $\mathbf{Q}$ and $\mathbf{K}$ values by scaling in Transformer-based architectures to satisfy $\mathbf{Q} = \mathbf{K}$ while preserving the $\mathbf{QK}^T$ product.

As illustrated in Figure 3, WISCA modifies the initialization by enforcing $\mathbf{Q} = \mathbf{K}$, leading to fundamentally different optimization dynamics compared to random initialization. SGD-M-WISCA exhibits smoother and more stable convergence, avoiding sharp minima with WISCA-adjusted initialization.

WISCA can be applied not only during initialization but also at any iteration of the training process. Since the WISCA-adjusted model remains functionally equivalent to the original model (Definition 1), training can seamlessly resume from the transformed parameters.

Next, we will introduce the proposed WISCA in detail. The design of WISCA starts with vanilla self-attention in Transformer. To ensure similarity between $\mathbf{Q}$ and $\mathbf{K}$ while preserving the $\mathbf{QK}^T$ product, we devise the following adjustment for W_q and W_k in WISCA:

$$W_q' = W_q \cdot \sqrt{\frac{\|W_k\|_1}{\|W_q\|_1}} \quad (14)$$

$$W_k' = W_k \cdot \sqrt{\frac{\|W_q\|_1}{\|W_k\|_1}} \quad (15)$$

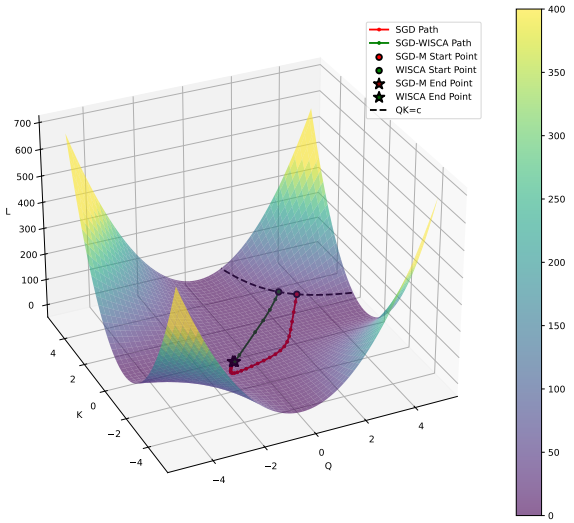

Figure 3: **Comparison of SGD-M Bottou and Bousquet (2007) and SGD-M-WISCA Optimization Paths.** **Red**: Optimization trajectory of SGD with momentum ($\beta = 0.9$) starting from a random initialization. **Green**: Trajectory using WISCA-adjusted initialization ($Q = K$). Both methods minimize $L = (QK - 1)^2$ with learning rate $\eta = 0.01$ and loss threshold $\epsilon = 10^{-2}$. **Observations**: 1. The WISCA path (**Green**) converges in **7 iterations** with a smooth, flat trajectory, while the random initialization (**Red**) requires **25 iterations** with oscillatory behavior. 2. All equivalent models on the same contour line ($QK = C$) exhibit slower convergence except at $Q = K$, where the flattest region enables faster stabilization. This demonstrates WISCA's ability to identify high-quality initializations aligned with flat minima, accelerating convergence without architectural changes.

where $\|\cdot\|_1$ denotes the L_1-norm of a matrix (sum of absolute values of its elements).

Why does the convergence become stable after using WISCA? Here is an intuitive explanation. Considering the landscape $f(Q, K) = (QK - 1)^2$ and the current checkpoint is (Q, K), the gradient is $2(QK - 1)(K, Q)$. After updating the parameter along the negative gradient direction with learning rate $\eta = \frac{\epsilon}{2(QK-1)}$, the next checkpoint is $(Q - \epsilon K, K - \epsilon Q)$. Now the gradient at this checkpoint is $2((Q-\epsilon K)(K-\epsilon Q)-1)(K-\epsilon Q, Q-\epsilon K)$. We hope the gradient direction varies as small as possible. Therefore, we have $\frac{Q}{K} = \frac{Q-\epsilon K}{K-\epsilon Q}$ and obtain $K^2 = Q^2$, which means $|Q| = |K|$.

The proposed scaling strategy is also valid for consecutive linear layers, even when activation functions (e.g., ReLU, LeakyReLU Xu et al. (2015)) are used. Similarly, it is also valid for w_v and w_o in transformer.

To ensure $\|W_v'\|_1 = \|W_o'\|_1$ while preserving the

output Output $= (attention\_score \cdot V) \cdot W_o$, we define:

$$W_v' = W_v \cdot \sqrt{\frac{\|W_o\|_1}{\|W_v\|_1}} \tag{16}$$

$$W_o' = W_o \cdot \sqrt{\frac{\|W_v\|_1}{\|W_o\|_1}} \tag{17}$$

In classical Transformer architectures, W_q and W_k typically share the same dimensionality. Under Gaussian initialization, their L_1/L_2-norms become approximately equal as the parameter count grows, rendering naive WISCA adjustments negligible (Appendix Theorem 2).

Another essential application of WISCA is for the GQA-based architectures Ainslie et al. (2023) (e.g., g query heads sharing one key/value group), W_q has $g\times$ more parameters than W_k, leading to a significant WISCA scaling ratio of $\sqrt{1/g}$. For GQA, we propose **group-averaged normalization**:

- **Loss Flatness Principle**: For $\mathcal{L} = (\mathbf{QK}-1)^2$, flatness is maximized when $\mathbf{Q} = \mathbf{K}$.
- **GQA Simulation**:

$$\begin{aligned}\mathbf{Q} &= [Q_1, Q_2, \ldots, Q_g],\\ \mathbf{K} &= [K_1, K_1, \ldots, K_1],\\ \mathcal{L} &= [K_1 \textstyle\sum_{i=1}^{g} Q_i - 1]^2\end{aligned}$$

  Optimal flatness occurs when $K_1 = \sum_{i=1}^{g} Q_i$, not $K_1 = Q_1 = Q_2 = \cdots = Q_g$.

Conclusion: WISCA adjustments for GQA enforce $\|W_q\|_1 = \|W_k\|_1 \times g$, yielding **larger scaling effects** than in MHA. Similarly, WISCA can be extended to matrix multiplication with unequal matrices, such as the A and B matrices of LoRA.

We have demonstrated both theoretically and empirically that WISCA achieves its goals through core principles in Transformer architecture: Adjust $\|W_q\|_1 = \|W_k\|_1$ and $\|W_v\|_1 = \|W_o\|_1$, while preserving model outputs. The impact of WISCA is more pronounced in architectures with imbalanced parameter groups, such as Grouped Query Attention (GQA), where scaling (e.g., $\sqrt{1/g}$) leads to significant tuning.

In experiments, we implement two WISCA variants:

- **Tensor-wise WISCA**: Global scaling applied to entire weight matrices (Equations 14–17).
- **Channel-wise WISCA**: A finer-grained variant that applies scaling at the channel level, enhancing flexibility in parameter optimization.

4 Experiments

We designed tensor-wise and channel-wise WISCA for W_q/W_k and W_v/W_o, and conducted pre-training comparative experiments on open-source Llama, Qwen, and popular Mixture-of-Experts (MoE) architectures (Jacobs et al., 1991). The experiments evaluated the training convergence and downstream task performance of different WISCA strategies compared to the original approach. Results demonstrate that WISCA significantly improves model training performance across all tested architectures.

4.1 Tensor-wise and Channel-wise WISCA

We implement four WISCA variants targeting different components of the attention mechanism: (1) adjustments on W_q/W_k for attention score computation, covering tensor-wise (Figure 4) and channel-wise (Figure 5) optimizations; and (2) adjustments on W_v/W_o for output projection, also including tensor-wise (Figure 6) and channel-wise (Figure 7) variants.

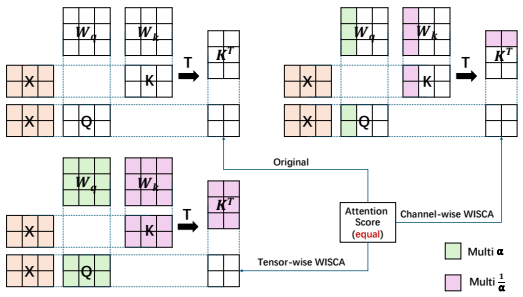

Figure 4: **Comparison of Attention Score Computation Methods.** Original attention mechanism (top-left), tensor-wise WISCA (bottom-down), and channel-wise WISCA (top-right) on W_q/W_k adjustments. All methods produce **identical attention scores** but exhibit **distinct weight patterns**.

As shown in Figure 4, both QK-WISCA variants retain the original attention scores while redistributing weight magnitudes, with channel-wise scaling offering finer-grained control than tensor-wise. Similarly, Figure 6 shows that W_v/W_o adjustments preserve self-attention outputs despite altered weight patterns, mirroring the pattern observed in attention-score optimization.

For Grouped Query Attention (GQA) architectures, we further validate the adaptability of

channel-wise WISCA. Figures 5 and 7 demonstrate that GQA's dimensional asymmetry requires group-aware scaling in channel-wise WISCA.

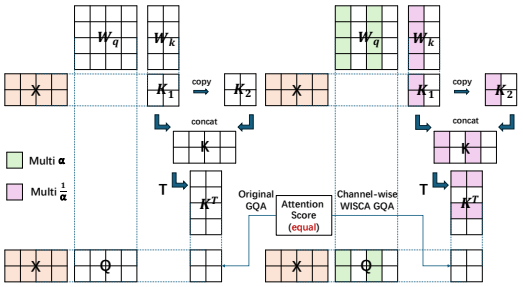

Figure 5: **Channel-wise WISCA in GQA Architecture.** Comparison between original attention (left) and channel-wise WISCA (right) for W_q/W_k adjustments in Grouped Query Attention (GQA). Both methods produce **identical attention scores** but exhibit **distinct weight patterns**. GQA introduces dimensional asymmetry between W_q and W_k, requiring channel-wise WISCA to adapt to group-wise scaling (tensor-wise WISCA remains identical to MHA, omitted here).

4.2 Convergence Effect on Mainstream LLMs

To validate WISCA's effectiveness in real training scenarios, we conduct pre-training experiments on three architectures (**TinyLlama** (Zhang et al., 2024), **Qwen2-1.5B** (Team, 2024b), and **Qwen1.5-MoE** (Team, 2024a)) using the TinyStories (Eldan and Li, 2023) dataset. We compare tensor-wise WISCA strategies (① QK_TEN and ② VO_TEN shown in Figure 4 and Figure 6) against the original training approach, with WISCA applied every 250 iterations while maintaining identical hyperparameters (learning rate, optimizer, batch size, etc.) across all experimental groups. Training loss and test perplexity (PPL) results are summarized in Table 1.

Table 1: Training losses and test PPL scores for different strategies on various models

Model	Strategy	Train Loss	Test PPL
TinyLlama	origin	1.3193	3.78
	① QK_TEN	1.2849	3.66
	② VO_TEN	1.3123	3.75
	①+②	**1.2749**	**3.62**
Qwen2-1.5B	origin	1.355	3.96
	① QK_TEN	1.3417	3.91
	② VO_TEN	1.3507	3.94
	①+②	**1.3336**	**3.88**
Qwen1.5-MoE	origin	1.5497	4.76
	① QK_TEN	1.519	4.62
	② VO_TEN	1.5408	4.72
	①+②	**1.5141**	**4.60**

Across all pre-training tasks, the experimental results demonstrate that WISCA consistently improves model convergence behavior. Notably, the combined application of QK-WISCA and VO-WISCA yields superior performance compared to individual module adjustments, exhibiting a **synergistic effect** on the target dataset. This improvement pattern remains consistent across all tested architectures.

4.3 Evaluation Results on LLMs

To evaluate WISCA's zero-shot generalization capabilities, we trained a 1.1B-parameter Llama model from scratch on 1.4B tokens from Wikipedia.en, followed by comprehensive evaluation using the EleutherAI LM Evaluation Harness (Gao et al., 2024).

Table 2: Evaluation Metrics of llama-1.1B Trained on 1.4B Tokens from Wikipedia.en. Performance comparison of different attention optimization methods. All metircs seen in table 6

version	BoolQ↑	ARC-c↑	PIQA↑	WinoG↑	avg↑
origin	0.3838	0.1741	0.5288	0.5004	0.3968
① qk_ten	0.3810	0.1843	0.5332	0.4807	0.3948
② qk_row	0.3887	0.1817	0.5370	0.5091	0.4041
③ vo_ten	0.4015	0.1852	0.5294	0.4957	0.4030
④ vo_row	0.3817	0.1749	0.5386	0.4988	0.3985
①+③	**0.5214**	0.1869	0.5413	0.4980	**0.4369**
②+③	0.4483	**0.2014**	0.5305	0.5059	0.4215
①+④	0.4226	0.1783	0.5310	0.5075	0.4099
②+④	0.3994	0.1724	**0.5430**	**0.5170**	0.4080
①+③(init)	0.4703	0.1860	0.5299	0.4964	0.4207
②+③(init)	0.3960	0.1792	0.5386	0.5036	0.4044
①+④(init)	0.4312	0.1766	0.5308	0.4949	0.4083
②+④(init)	0.3917	0.1817	0.5337	0.5043	0.4029

In the end-to-end evaluation experiments, we validated both tensor/channel-wise WISCA strategies and their four combinatorial variants. Given the pronounced impact of initial WISCA adjustments on model positioning within the loss landscape, we additionally evaluated strategies where WISCA was applied only at initialization (marked as "(init)" in Table 2).

The evaluation experiments demonstrate that combinatorial WISCA strategies (①+③, etc.) significantly outperform individual optimizations, with the best combination achieving a 10.1% average improvement over the baseline (Table 2), highlighting synergistic effects between QK-WISCA and VO-WISCA adjustments.

While initialization-only WISCA retains an average of 97% of the full combinatorial performance,

it suggests that periodic adjustments provide incremental refinements to weight patterns. For resource-constrained scenarios, initialization-only WISCA offers a computationally efficient alternative (no training overhead) while preserving most performance gains.

Table 3: Zero-shot evaluation results on different training steps on llama_moe-5B-A0.8B.All metircs seen in table 7

Steps	Metrics AVG↑			PPL(wikitext2)		
(Tokens)	origin	①+③	+(%)	origin	①+③	+(%)
1.43B	35.56	37.07	**4.25**	70.93	68.11	**3.98**
2.86B	37.88	38.23	**0.92**	48.17	46.25	**3.98**
4.29B	39.28	39.49	**0.53**	40.72	39.17	**3.82**
5.72B	39.83	40.68	**2.13**	35.48	34.85	**1.77**
7.15B	40.13	40.33	**0.50**	33.29	32.77	**1.56**
8.58B	40.31	40.69	**0.94**	31.16	30.80	**1.15**
10.0B	39.39	40.72	**3.38**	29.45	29.21	**0.82**

To analyze WISCA's impact across training stages, we train a Llama-MoE-5B-A0.8B model for 10B tokens, evaluating seven intermediate checkpoints. Zero-shot performance metrics and perplexity scores are reported in Table 3, demonstrating WISCA's consistent optimization benefits throughout training.

The results demonstrate that, due to the extensive tuning of WISCA during initialization, the most significant improvements are achieved early in training, when the model's loss graph smoothness undergoes the largest shift.

While metric fluctuations occur across training steps (e.g., 0.50–3.38% accuracy variations), WISCA consistently outperforms the baseline in all evaluations. The sustained positive trends—particularly the average 1.81% metrics improvement and 2.44% perplexity reduction across 10B tokens—validate its robustness as a general optimization strategy.

4.4 Benefits on LoRA

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is a parameter-efficient fine-tuning (PEFT) paradigm widely adopted for large language models (LLMs). The core hypothesis posits that weight updates (ΔW) during model adaptation exhibit a low *intrinsic rank*, implying they can be approximated by low-dimensional structures rather than full-rank matrices.

Given a pre-trained weight matrix $W \in \mathbb{R}^{m\times n}$, LoRA introduces two trainable low-rank matrices $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times n}$, where $r \ll \min(m, n)$. The adapted weight is computed as:

$$W' = W + \Delta W = W + A \cdot B \tag{18}$$

During fine-tuning, *only* A and B are updated, while the original W remains frozen.

When $m \neq n$, the parameter counts of matrices A and B become imbalanced. In standard LoRA, the B matrix is initialized as zero ($\|A\| \neq \|B\|$), enabling dynamic WISCA operations during training. We fine-tuned the DeepSeek-V2-Lite-Chat model on the Alpaca and MetaMath datasets using LoRA, PISSA (Meng et al., 2024), and RS-LoRA (Kalajdzievski, 2023) strategies via Llama-Factory (Zheng et al., 2024), comparing vanilla fine-tuning with WISCA-enhanced training (Table 4). Results validate its universal optimization capability in parameter-efficient fine-tuning (PEFT) scenarios.

The **Alpaca** (Taori et al., 2023) dataset is a widely-used benchmark for instruction tuning, generated by fine-tuning LLaMA on self-instruct-style human-AI interactions to improve language models' ability to follow diverse commands. **MetaMath** (Yu et al., 2023) is a specialized mathematical reasoning dataset containing 8.4k problem-solution pairs across arithmetic, algebra, and geometry, designed to evaluate and enhance models' multistep reasoning capabilities.

Table 4: Training Loss Comparison: Vanilla vs. WISCA enhanced SFT fine-tuning for DeepSeek-V2-Lite-Chat (DeepSeek-AI, 2024) on Alpaca and MetaMath. All experiments were trained with 3 epochs.

Method	Alpaca		MetaMath	
	Origin	WISCA	Origin	WISCA
LoRA	0.8602	**0.8532**	0.0779	**0.0770**
PISSA	0.6227	**0.6176**	0.0692	**0.0688**
RS-LoRA	0.6773	**0.6760**	0.0744	**0.0726**

Since the B matrix in vanilla LoRA is initialized as a zero matrix, WISCA cannot be applied during initialization. In our experiments (Table 4), WISCA operations were performed at one-third of the total training iterations.

4.5 Benefits on Eagle

EAGLE (Li et al., 2024) is a state-of-the-art speculative sampling framework that significantly accelerates LLMs' inference. It employs a draft model (typically a small neural network) to generate candidate token sequences, which are then validated in parallel by the target LLM.

We implement EAGLE by training a draft model for **LLaMA 3.1 Instruct 8B** (Grattafiori et al., 2024). The draft model architecture consists of a single Transformer layer with self-attention, incorporating our proposed WISCA strategy during training to optimize weight patterns.

Dataset: We use the **ShareGPT** dataset, a collection of 90K high-quality human-AI conversations spanning diverse domains (e.g., coding, reasoning, open-ended dialogue). Each example contains multi-turn interactions, making it suitable for training instruction-following draft models.

Table 5: EAGLE-1 Performance: Average Accepted Tokens and Training Metrics. Tree/Chain modes use depth=6/top-k=10 and depth=5, respectively.

Metric	Original		WISCA	
	Tree	Chain	Tree	Chain
MT-Bench	3.001	1.528	**3.009**	**1.593**
GSM8K	3.234	**1.818**	**3.271**	1.809
HumanEval	3.708	2.103	**3.715**	**2.110**
Alpaca-Eval	2.874	1.462	**2.900**	**1.478**
Train Loss	0.7168		**0.7113**	
Val Loss	0.7359		**0.7312**	
Train Acc (%)	79.28		**79.49**	
Val Acc (%)	77.03		**77.11**	

In experiments with the EAGLE draft model (Table 5), we applied tensor-wise WISCA to the $[w_q, w_k]$ and $[w_v, w_o]$ modules. The results show that WISCA consistently improves training efficiency, with the WISCA-enhanced model outperforming the baseline in end-to-end evaluations in both *tree* and *chain* generation modes. This validates the generalizability of WISCA's weight pattern optimization in speculative sampling frameworks.

5 Related Work

Loss Landscape Optimization. Sharpness-Aware Minimization (SAM) (Foret et al., 2020) explicitly minimizes loss sharpness, albeit at the cost of approximately double the computation per parameter update. Stochastic Weight Averaging (SWA) (Izmailov et al., 2018) improves generalization by weight averaging, but requires extensive training. WISCA implicitly smooths the loss landscape through weight pattern adjustments , achieving efficiency gains with minimal overhead.

Weight Balancing and Normalization. Query key normalization (QKN) (Henry et al., 2020) improves the stability of transformer training by applying L2 normalization to the query (Q) and key (K) matrices, transforming their dot products into cosine similarities. Weight normalization (Salimans and Kingma, 2016) reparameterizes weights to decouple direction and magnitude. Unlike these methods, WISCA enforces *functional equivalence* while balancing weight norms, enabling dynamic transitions between equivalent models without architectural changes.

Model Equivalence and Weight Patterns. The Lottery ticket hypothesis (Frankle and Carbin, 2018) identifies performant sparse subnets, whereas permutation symmetry reveals equivalence under weight permutations. WISCA leverages these insights to navigate the loss landscape via equivalent model transitions, prioritizing flat minima for enhanced generalization.

Parameter-Efficient Fine-Tuning (PEFT). Low-Rank Adaptation (LoRA) (Hu et al., 2022) reduces trainable parameters by decomposing weight updates into low-rank matrices. Recent variants like PiSSA (Meng et al., 2024) further improve efficiency by adapting only the principal singular values and vectors of the weight matrices, achieving faster convergence than LoRA. WISCA can be integrated with PEFT methods such as LoRA and PiSSA to optimize weight patterns in their low-rank matrices, enhancing both training stability and task-specific adaptation without introducing additional parameters.

Speculative Decoding. EAGLE (Li et al., 2024) accelerates LLM inference through a small draft model. Our work improves EAGLE by applying WISCA to the draft layer training, improving token acceptance rates through better weight patterns.

6 Conclusion

In this work, we introduced WISCA, a training optimization strategy that reshapes weight patterns to enhance model robustness without architectural changes. WISCA improves generalization by achieving flatter minima through equivalent model transitions. Experiments across various architectures, such as GQA, MoE, and LoRA, demonstrated WISCA's effectiveness in reducing training perplexity, enhancing zero-shot performance, and increasing inference efficiency. This approach suggests potential for broader application and future exploration of equivalent model strategies across different neural networks.

7 Limitations

Despite the promising results, WISCA has several limitations that warrant further investigation:

Architectural Constraints: While WISCA effectively optimizes weight patterns in Transformer-based architectures, its applicability to other neural network structures, such as convolutional or recurrent networks, remains unexplored. Further research is needed to adapt WISCA to diverse architectures.

Experimental Scope: The experiments conducted focus primarily on specific datasets and model configurations. The generalizability of WISCA across different data domains and larger-scale models requires comprehensive evaluation.

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

A.R. Barron. 1993. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Léon Bottou and Olivier Bousquet. 2007. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*, 9.

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

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

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. 2020. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*.

Feiye Huo, Jianchao Tan, Kefeng Zhang, Xunliang Cai, and Shengli Sun. 2025. C2t: A classifier-based tree construction method in speculative decoding. *arXiv preprint arXiv:2502.13652*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 2002. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Jiayu Qin, Jianchao Tan, Kefeng Zhang, Xunliang Cai, and Wei Wang. 2025. Maskprune: Mask-based llm pruning for layer-wise uniform structures. *arXiv preprint arXiv:2502.14008*.

Tim Salimans and Durk P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, and 1 others. 2025. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*.

Qwen Team. 2024a. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters".

Qwen Team. 2024b. Qwen2 technical report. *arXiv preprint arXiv:2412.15115*.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *CVPR*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Norm Convergence in Classical Transformers

Theorem 2. *For Gaussian-initialized matrices $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{m\times n}$ with i.i.d. entries $\mathcal{N}(0,\sigma^2)$, the L_1- and L_2-norms satisfy:*

$$\frac{\|\mathbf{W}_q\|_1}{\|\mathbf{W}_k\|_1} \xrightarrow{p} 1 \quad \textit{and} \quad \frac{\|\mathbf{W}_q\|_2}{\|\mathbf{W}_k\|_2} \xrightarrow{p} 1 \quad \textit{as } mn \to \infty,$$

where $\xrightarrow{p}$ denotes convergence in probability.

Proof. 1. **L_2-Norm (Frobenius Norm)**: Let $\|\mathbf{W}\|_2^2 = \sum_{i,j} W_{ij}^2$. For Gaussian entries:

$$\mathbb{E}[\|\mathbf{W}\|_2^2] = mn\sigma^2, \quad \text{Var}(\|\mathbf{W}\|_2^2) = 2mn\sigma^4.$$

By Chebyshev's inequality:

$$P\left(\left|\frac{\|\mathbf{W}\|_2^2}{mn\sigma^2} - 1\right| > \epsilon\right) \le \frac{2mn\sigma^4}{(mn\sigma^2\epsilon)^2} = \frac{2}{\epsilon^2 mn} \to 0.$$

Thus, $\|\mathbf{W}_q\|_2/\|\mathbf{W}_k\|_2 \to 1$.

2. **L_1-Norm**: Let $\|\mathbf{W}\|_1 = \sum_{i,j} |W_{ij}|$. For $W_{ij} \sim \mathcal{N}(0,\sigma^2)$:

$$\mathbb{E}[|W_{ij}|] = \sigma\sqrt{\frac{2}{\pi}}, \quad \text{Var}(|W_{ij}|) = \sigma^2\left(1 - \frac{2}{\pi}\right).$$

Summing over mn terms:

$$\mathbb{E}[\|\mathbf{W}\|_1] = mn\sigma\sqrt{\frac{2}{\pi}},$$
$$\text{Var}(\|\mathbf{W}\|_1) = mn\sigma^2\left(1 - \frac{2}{\pi}\right).$$

Again by Chebyshev:

$$P\left(\left|\frac{\|\mathbf{W}\|_1}{mn\sigma\sqrt{2/\pi}} - 1\right| > \epsilon\right) \le \frac{1 - 2/\pi}{\epsilon^2 mn} \to 0.$$

Hence, $\|\mathbf{W}_q\|_1/\|\mathbf{W}_k\|_1 \to 1$. □

B Appendix figures

In this Appendix, we present the WISCA strategy for the `wv` and `wo` components of the transformer block, which were not exhibited in the main text. Figure 6 illustrates the matrix perspective of tensor-wise WISCA and channel-wise WISCA for a general attention module. Similarly, Figure 7 provides a schematic representation for the attention module with GQA.

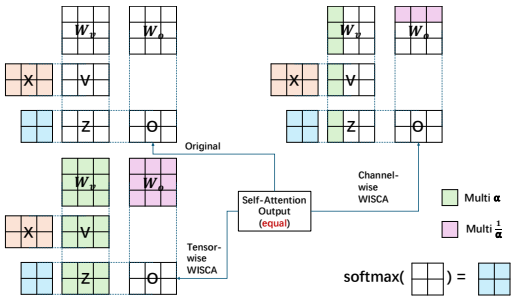

Figure 6: **Comparison of Output Projection Methods.** Original output projection (left), tensor-wise WISCA (middle), and channel-wise WISCA (right) on W_v/W_o adjustments. All methods produce **identical self-attention outputs** but exhibit **distinct weight patterns**.

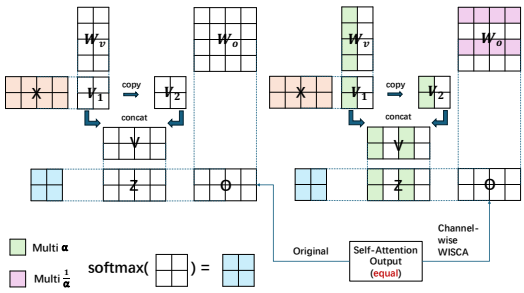

Figure 7: **Channel-wise WISCA for Output Projection in GQA.** Comparison between original output projection (left) and channel-wise WISCA (right) on W_V/W_O in Grouped Query Attention (GQA). Both methods yield **identical self-attention outputs** but with **distinct weight patterns**. GQA's dimensional asymmetry requires channel-wise WISCA to adapt group-aware scaling, while tensor-wise WISCA remains equivalent to MHA (omitted here).

C Summary of All Metrics

In the experimental section, we conducted pre-training tasks using the tinyllama and llama-moe models. While the main text includes a streamlined table focusing on zero-shot evaluation metrics, this appendix provides a complete overview of all evaluation metrics for thorough analysis.

The following tables provide detailed metrics gathered from our comprehensive evaluations. Table 6 breaks down the individual performance metrics across various tests, highlighting the comparative effectiveness of different attention optimization strategies applied to the Llama model. We additionally explore the impact of WISCA at various stages of training in Table 7, showcasing the zero-shot evaluation results for the llama_moe-5B-A0.8B model across different checkpoints and training steps.

These results underscore the efficacy of our approaches. By presenting all metrics here, we aim to furnish a deeper understanding of the evaluation outcomes in the context of our experimental framework.

C.1 Proof of Convergence

Why does the convergence become stable after using WISCA? Here is an intuitive explanation. Considering the landscape $f(Q,K) = (QK-1)^2$ and the current checkpoint is (Q,K), the gradient is $2(QK-1)(K,Q)$. After updating the parameter along the negative gradient direction with learning rate $\eta = \frac{\epsilon}{2(QK-1)}$, the next checkpoint is $(Q-\epsilon K, K-\epsilon Q)$. Now the gradient at this checkpoint is $2((Q-\epsilon K)(K-\epsilon Q)-1)(K-\epsilon Q, Q-\epsilon K)$. We hope the gradient direction varies as small as possible. Therefore, we have $\frac{Q}{K} = \frac{Q-\epsilon K}{K-\epsilon Q}$ and obtain $K^2 = Q^2$, which means $|Q| = |K|$.

Here are the details for above. Consider the minimal analytic example:

$$L(Q,K) = \tfrac{1}{2}(QK-C)^2, \qquad C > 0.$$

a) Hessian and sharpness The Hessian and its trace are

$$\mathbf{H} = \begin{bmatrix} K^2 & QK \\ QK & Q^2 \end{bmatrix}, \qquad \mathrm{Tr}(\mathbf{H}) = Q^2 + K^2.$$

On the contour $QK = C$, $\mathrm{Tr}(\mathbf{H})$ is minimized when $|Q| = |K| = \sqrt{C}$, giving the flattest region.

b) One-step SGD stability Update rules:

$$Q_1 = Q_0 - \eta\,(Q_0K_0 - C)\,K_0$$

$$K_1 = K_0 - \eta\,(Q_0K_0 - C)\,Q_0$$

Let $\mathbf{g}^{(1)} = [K_1;\, Q_1]$ be the next gradient direction. A first-order expansion gives

$$\cos\theta = 1 - \eta\,(Q_0K_0 - C)(Q_0^2 - K_0^2)^2/||g||^2 + O(\eta^2)$$

Hence $\cos\theta = 1$ if and only if $|Q_0| = |K_0|$, i.e., no directional oscillation.

c) Guaranteed sharpness reduction After rescaling to $|Q'| = |K'| = \sqrt{C}$,

$$\Delta\mathrm{Tr}(\mathbf{H}) = Q_0^2 + K_0^2 - 2C = (|Q_0| - |K_0|)^2 \geqslant 0$$

which guarantees a deterministic drop of curvature before any gradient step.

Table 6: All metrics of table2

version	BoolQ↑	ARC-c↑	ARC-e↑	PIQA↑	WinoG↑	HellaS↑	OBQA↑	ceval↑	Avg↑
origin	0.3838	0.1741	**0.3274**	0.5288	0.5004	0.2650	0.1300	0.2437	0.3192
① qk_ten	0.3810	0.1843	0.3228	0.5332	0.4807	0.2617	0.1280	0.2511	0.3179
② qk_row	0.3887	0.1817	0.3102	0.5370	0.5091	0.2617	0.1280	0.2533	0.3212
③ vo_ten	0.4015	0.1852	0.3081	0.5294	0.4957	0.2653	**0.1380**	0.2585	0.3227
④ vo_row	0.3817	0.1749	0.3178	0.5386	0.4988	0.2629	0.1500	0.2548	0.3224
①+③	**0.5214**	0.1869	0.3165	0.5413	0.4980	0.2634	0.1340	0.2585	**0.3400**
②+③	0.4483	**0.2014**	0.3165	0.5305	0.5059	**0.2656**	0.1320	0.2578	0.3323
①+④	0.4226	0.1783	0.3199	0.5310	0.5075	0.2645	0.1260	0.2377	0.3234
②+④	0.3994	0.1724	0.3224	**0.5430**	**0.5170**	0.2621	0.1340	0.2377	0.3235
①+③(init)	0.4703	0.1860	0.3182	0.5299	0.4964	0.2637	0.1300	**0.2623**	0.3321
②+③(init)	0.3960	0.1792	0.3207	0.5386	0.5036	0.2652	0.1120	0.2355	0.3189
①+④(init)	0.4312	0.1766	0.3203	0.5308	0.4949	0.2644	0.1220	0.2541	0.3243
②+④(init)	0.3917	0.1817	0.3199	0.5337	0.5043	0.2631	0.1240	0.2311	0.3187

Table 7: Zero-shot evaluation results on different training steps on llama_moe-5B-A0.8B.

Metric	1w	2w	3w	4w	5w	6w	7w
Trained Tokens	1.43B	2.86B	4.29B	5.72B	7.15B	8.58B	10B
Baseline							
ARC-c	19.28	18.69	17.66	19.11	20.22	19.20	20.73
ARC-e	32.74	37.54	41.33	42.80	43.43	43.56	44.57
BoolQ	48.10	60.06	60.55	62.05	61.13	61.07	49.36
H-Swag	26.80	27.32	27.86	28.47	28.97	29.44	29.84
OpenBQ	11.80	12.80	13.80	13.40	14.00	16.00	15.60
PIQA	57.73	58.60	60.39	61.53	62.40	63.11	62.89
Wino	52.49	50.12	53.35	51.46	50.75	49.80	52.72
AVG	35.56	37.88	39.28	39.83	40.13	40.31	39.39
PPL(wikitext2)	70.93	48.17	40.72	35.48	33.29	31.16	29.45
QK							
ARC-c	18.26	18.43	18.60	17.83	18.77	18.34	20.22
ARC-e	33.00	37.58	39.86	42.68	44.19	45.29	44.78
BoolQ	49.11	62.23	62.08	62.17	61.87	61.80	61.41
H-Swag	26.65	27.34	28.17	28.57	28.95	29.43	29.74
OpenBQ	12.00	14.00	14.60	15.00	16.00	15.00	14.60
PIQA	57.13	59.63	60.88	62.35	63.00	63.66	64.31
Wino	50.67	52.41	50.75	51.85	50.51	52.88	52.49
AVG	35.26	38.80	39.28	40.06	40.47	40.91	41.08
Metric.Gain(%)	-0.84	**+2.43**	0.00	**+0.58**	**+0.85**	**+1.49**	**+4.29**
PPL(wikitext2)	69.44	47.12	39.74	34.97	32.86	30.99	29.19
PPL.Gain(%)	**+2.10**	**+2.19**	**+2.42**	**+1.43**	**+1.29**	**+0.54**	**+0.85**
QKVO							
ARC-c	17.58	18.60	18.09	19.97	18.77	20.39	18.77
ARC-e	35.40	38.22	40.74	43.77	43.94	44.82	44.19
BoolQ	60.70	61.80	60.09	61.74	61.68	61.80	59.79
H-Swag	26.75	27.49	28.12	28.69	28.89	29.40	29.89
OpenBQ	11.40	13.00	16.40	15.60	16.40	15.40	17.80
PIQA	58.38	59.41	61.26	62.68	62.62	63.11	63.60
Wino	49.25	49.09	51.70	52.33	50.04	49.88	50.99
AVG	37.07	38.23	39.49	40.68	40.33	40.69	40.72
Metric.Gain(%)	**+4.25**	**+0.92**	**+0.53**	**+2.13**	**+0.50**	**+0.94**	**+3.38**
PPL(wikitext2)	68.11	46.25	39.17	34.85	32.77	30.80	29.21
PPL.Gain(%)	**+3.98**	**+3.98**	**+3.82**	**+1.77**	**+1.56**	**+1.15**	**+0.82**