

**Organization** In this Appendix, we provide in-depth descriptions of the materials that are not covered in the main paper, and report additional experimental results. The document is organized as follows:

- **A-** Related Work
- **B-** Theoretical Supplement
  - **B.1** Theoretical Justification of Angle-Dependent Effects in LLMs
  - **B.2** Attention-Based Explanation of Layer-wise Angle Concentration Patterns
  - **B.3** Neuron-Based Explanation of Data-wise Angle Concentration Patterns
- **C-** Visualization Results
  - **C.1** Layer-wise Angle Concentration Patterns
  - **C.2** Data-wise Angle Concentration Patterns
- **D-** GAIN-RL Algorithm
- **E-** Experimental Setup
- **F-** Additional Experimental Results
  - **F.1** Performance of Weighted Signals
  - **F.2** Model Performance on Single Task
- **G-** Discussion and Future Work

## **A Related Work**

Reinforcement Fine-Tuning (RFT)[1, 2] has demonstrated significant effectiveness in enhancing the reasoning capabilities of large language models. However, despite its promising potential, its low sample efficiency and high computational costs remain critical barriers to the broader adoption of RFT. Recent efforts aimed at addressing these efficiency have primarily focused on algorithmic optimizations and data-centric strategies. Algorithmic optimizations, exemplified by methods such as GRPO [1], REINFORCE++ [2], and ReMax [3], seek to reduce computational complexity by streamlining underlying RL algorithms. Although these methods typically improve efficiency and stability, they often involve inherent trade-offs. For instance, GRPO estimates advantages through relative comparisons within output groups, thereby eliminating the need for value functions. While this approach reduces complexity and the reliance on critics, it can introduce instability due to increased noise in advantage estimation, higher variance in updates, and greater sample requirements.

In parallel, data-centric strategies [4, 5, 6, 7, 8, 9] have emerged as promising alternatives for efficient fine-tuning. For example, [4] provides the first quantitative audit of preference datasets, introducing metrics for scale, noise, and information density that expose quality bottlenecks before any policy update. Direct Preference Optimization (DPO) [10] simplifies the entire loop by replacing on-policy RL with a closed-form classification loss, making the choice of high-value preference data the primary driver of alignment quality. To further reduce annotation cost, Active Preference Optimization [11] casts RLHF as an active-learning bandit, adaptively querying only the prompts expected to maximize reward-model improvement. The latest data-centric strategies can be categorized based on their approach to data manipulation: data selection and data sequencing. Data selection techniques involve filtering extensive datasets to retain only a small subset of high-quality training data based on predefined metrics. Methods such as LIMO[12] and s1[13] have demonstrated that carefully curated small supervised fine-tuning datasets can achieve robust performance using orders of magnitude less data. On the other hand, data sequencing strategies[14] enhance model learning speed by rearranging the order of training data within existing datasets. Approaches like ADARFT[15] have shown that dynamically selecting data for each epoch can effectively accelerate the training process.

Nevertheless, existing data-centric strategies have generally failed to account for the unique characteristics of different models, applying uniform data handling procedures across diverse model architectures. Such uniformity can lead to suboptimal outcomes because models differ significantly in their sensitivity and response to the same datasets. To overcome this challenge, this paper aims to identify intrinsic signals within models that can reflect their perceptual capabilities toward data, thereby enabling tailored data strategies without incurring substantial additional costs.

## 50 B Theoretical Supplement

51 In this section, we provide the theoretical explanation supporting the main text in Section 2. Specifi-  
 52 cally, we elaborate on the Angle-Dependent Effects of Attention and Activation (corresponding to  
 53 Section 2.1 of the main text), the Attention-Based Explanation of Layer-wise Angle Concentration  
 54 Patterns (corresponding to Section 2.2 of the main text), and the Neuron-Based Explanation of  
 55 Data-wise Angle Concentration Patterns (corresponding to Section 2.3 of the main text).

### 56 B.1 Theoretical Justification of Angle-Dependent Effects in LLMs

57 In this section, we demonstrate through derivation of the LLM computational process that the nonlin-  
 58 ear operations in LLMs—namely attention and activation computations—are inherently dependent  
 59 on the angles between the input hidden states.

60 To support this claim, we first introduce two empirical assumptions based on observation:

61 *Observation 1.*  $W_q$  and  $W_k$  are nearly approximately orthogonal to each other, i.e.,  $W_q W_k^T \approx \theta I$ .  
 62  $\theta$  is a constant.  $(W_o, W_u, W_a)$  are approximately orthogonal matrices, i.e.,  $W W^T \approx \lambda I$ .  $\lambda$  is a  
 63 constant.

64 *Observation 2.* For activation function output vectors  $A_i$  and  $A_M$ ,  $\cos(\angle(A_i, A_M))$  is proportional  
 65 to the number of intersections of activated neurons, i.e.,  $|\Gamma(x_i) \cap \Gamma(x_M)|$ .

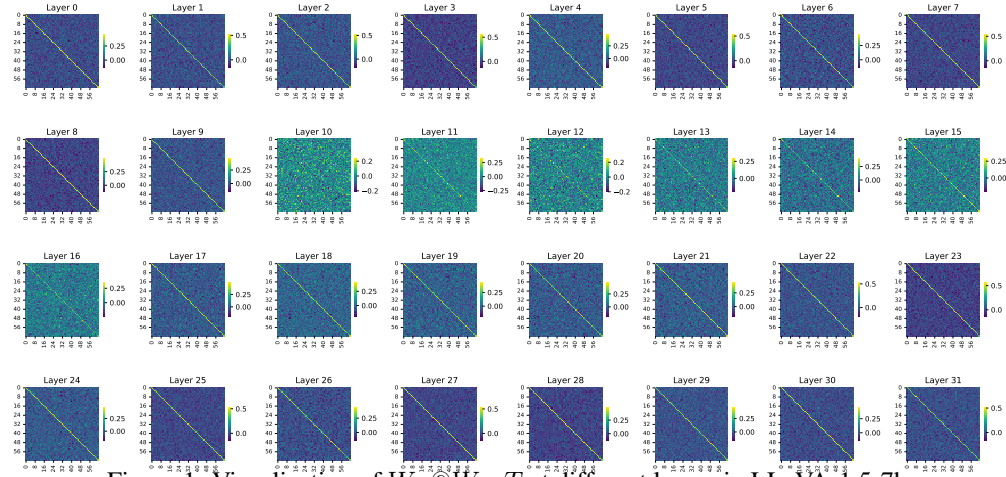


Figure 1: Visualization of  $W_Q @ W_K.T$  at different layers in LLaVA-1.5-7b.

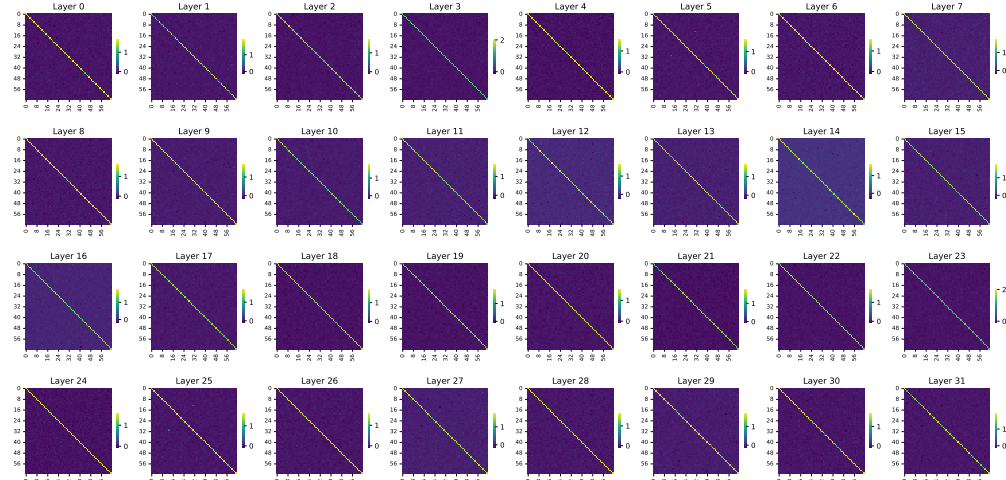


Figure 2: Visualization of  $W_D @ W_D.T$  at different layers in LLaVA-1.5-7b.

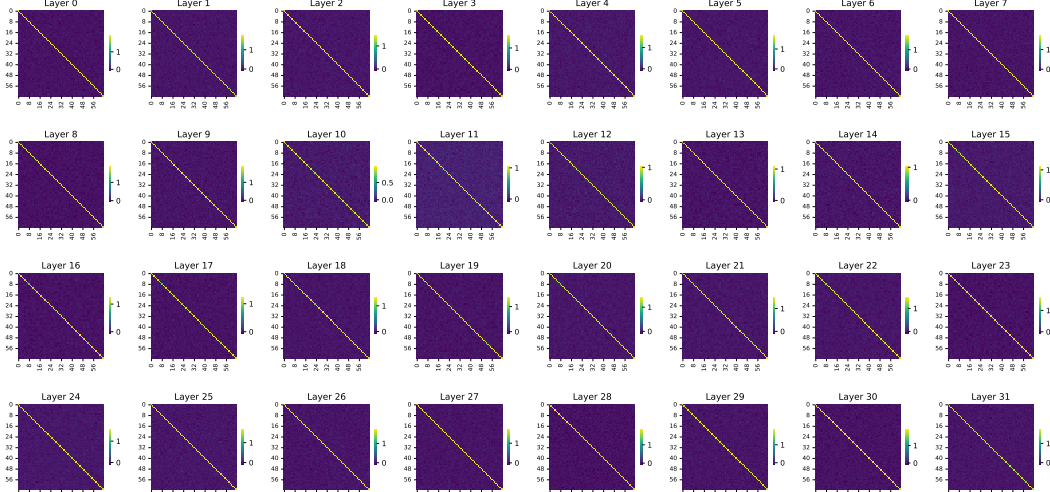


Figure 3: Visualization of  $W_V @ W_V.T$  at different layers in LLaVA-1.5-7b.

66 We provide empirical validation to support both of these assumptions.

67 For **Observation 1**, we show  $W_Q @ W_K.T$ ,  $W_V @ W_V.T$ ,  $W_D @ W_D.T$  of all different layers of

68 LLaVA-1.5-7B in Fig. 1, 2, and 3. It can be seen that different layers have this orthogonal relationship.

69 In fact, the orthogonal relationship of matrices in neural networks has been studied since a long time

70 ago. In particular, [16] proposed a new regularization method that encourages the weight matrix

71 of the neural network to maintain orthogonality during training by introducing a self-orthogonality

72 module. This method helps to improve the training stability and generalization ability of the model.

73 [17] explores adding orthogonal regularization to weights during training to improve training stability.

74 The author proposed an orthogonal regularization method for weights, aiming to solve the gradient

75 vanishing and explosion problems encountered by deep convolutional neural networks during training.

76 It can be seen that modules with orthogonality are found in various different models to improve

77 the training stability and performance of the model. To the best of our knowledge, we are the first

78 work to intuitively show this orthogonal performance in LLM, which can be more fully explored in

79 subsequent research.

80 For **Observation 2**, this observation is illustrated in Fig. 4. An intuitive understanding is that if  $x_i$

81 and  $x_M$  activate more of the same neurons,  $A_i$  and  $A_M$  will have more positive values in common

82 positions, making  $\cos(A_i, A_M)$  larger.

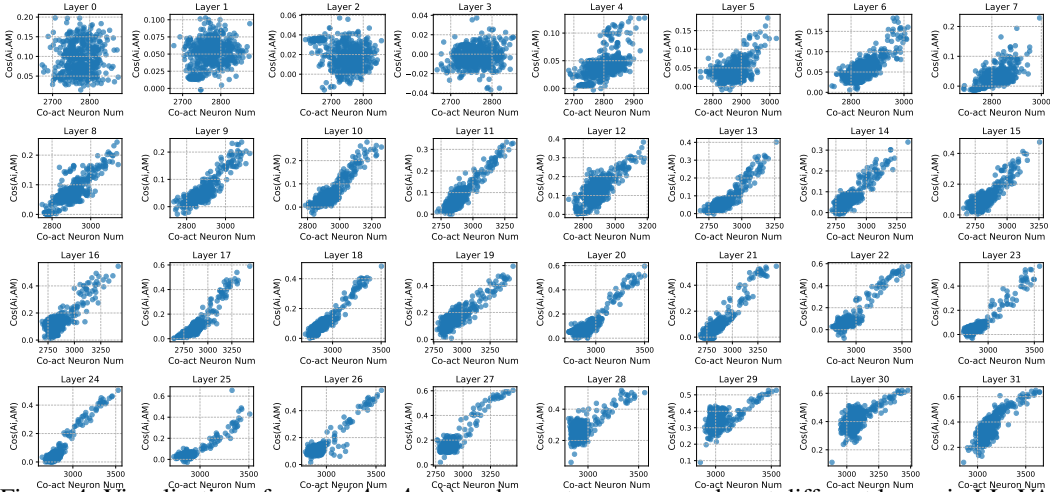


Figure 4: Visualization of  $\cos(\langle A_i, A_M \rangle)$  and co-act neurons number at different layers in LLaVA-1.5-7b.

83 *Theoretical Insight 1. Within an attention block, the degree of interaction between two tokens is*  
 84 *governed by the relative angle of their input hidden states.*

85 *Justification.* For a single token, suppose its input to the Attention block is  $y$ . Consider a sequence of  
 86 tokens  $[y_1, y_2, \dots, y_M]$ , for token  $y_m$ , its computation in the Attention layer can be expressed as:

$$\begin{aligned}\hat{y}_m &= \text{LayerNorm}(y_m), \quad V_m = \hat{y}_m W_v, \\ \alpha_{im} &= \text{Softmax}((\hat{y}_i W_q)(\hat{y}_m W_k)^T), \quad i < m \\ O_m &= \alpha_{1m} V_1 + \alpha_{2m} V_2 + \dots + \alpha_{mm} V_m,\end{aligned}\tag{1}$$

87 where  $\alpha_{im}$  is the attention score between the  $i$ -th and the  $m$ -th token.  $O_m$  is the output vector of the  
 88  $m$ -th token. To examine the influence of the  $i$ -th token  $y_i$  on the final output  $O_m$  of the  $m$ -th token,  
 89 we can consider the following projection value:

$$\|\text{Proj}_{O_m}(\alpha_{im} V_i)\| = \|\alpha_{im} V_i\| \cos(\angle(V_i, O_m)).\tag{2}$$

90 where  $\|\text{Proj}_{O_m}(\alpha_{im} V_i)\|$  is the projection value of  $\alpha_{im} V_i$  on  $O_i$ . And  $\cos(\angle(V_i, O_m))$  is the  
 91 cosine value of the angle between the  $V_i$  and  $O_m$  vectors. From Eq. 1,  $O_m$  is a sum of vectors in  
 92 different directions. Since the self-attention score  $\alpha_{mm}$  is typically much higher than  $\alpha_{im}$  for other  
 93 tokens, we can simplify the projection by assuming that  $O_m$  is primarily determined by  $\alpha_{mm} V_m$ , i.e.,

$$\|\text{Proj}_{O_m}(\alpha_{im} V_i)\| \approx \|\alpha_{im} V_i\| \cos(\angle(V_i, V_m)),\tag{3}$$

94 where  $\alpha_{im} = \text{Softmax}((\hat{y}_i W_q)(\hat{y}_m W_k)^T / \sqrt{d})$ . Since the Softmax function is monotonic, that is,  
 95  $\text{Softmax}(x) \propto x$ . We can have  $\alpha_{im} \propto (\hat{y}_i W_q)(\hat{y}_m W_k)^T$ . Therefore,

$$\begin{aligned}\|\alpha_{im} V_i\| \cos(\angle(V_i, V_m)) &\propto (\hat{y}_i W_q)(\hat{y}_m W_k)^T \|V_i\| \cos(\angle(V_i, V_m)) \\ &= \langle \hat{y}_i W_q, \hat{y}_m W_k \rangle \langle V_i, V_m \rangle / \|V_m\| \\ &= (\hat{y}_i (W_q W_k^T) \hat{y}_m^T) (\hat{y}_i (W_v W_v^T) \hat{y}_m^T) / \|V_m\|\end{aligned}\tag{4}$$

96 Based on the Observation 1,  $W_q W_k^T \approx \theta I$ ,  $W_v W_v^T \approx \lambda I$ . Therefore, combining Eq. 3 and Eq. 4,

$$\|\text{Proj}_{O_m}(\alpha_{im} V_i)\| \propto \theta \lambda \langle \hat{y}_i, \hat{y}_m \rangle \langle \hat{y}_i, \hat{y}_m \rangle / \|V_m\|\tag{5}$$

97 Given that  $\hat{y}$  is the result of  $y$  after LayerNorm,  $\hat{y}$  and  $y$  share the same direction, and  $\|\hat{y}\| = 1$ , it  
 98 holds that  $\langle \hat{y}_i, \hat{y}_m \rangle = \cos(\angle(y_i, y_m))$ . We can have

$$\|\text{Proj}_{O_m}(\alpha_{im} V_i)\| \propto \cos(\angle(y_i, y_m)),\tag{6}$$

99 Eq. 9 shows that the angle between different tokens directly affects their mutual interaction in attention  
 100 layer. The closer the angles of two tokens, the greater their mutual influence.  $\square$

101 We also demonstrate that the computation of activations is influenced by the angular relationships  
 102 between hidden states. This justification is presented as Theoretical Insight 4 in Section B.3.

## 103 B.2 Attention-Based Explanation of Layer-wise Angle Concentration Patterns

104 In this section, we present an attention-based explanation of Layer-wise angular concentration  
 105 patterns. Specifically, we theoretically show that: (1) the degree of angular concentration between  
 106 two hidden states influences the magnitude of their attention scores, and (2) the presence of sink  
 107 attention structure encourages angular concentration among tokens.

108 *Theoretical Insight 2. The smaller the angle between the input hidden states of two tokens to the*  
 109 *attention block, the higher their corresponding attention score.*

110 *Justification.* For two tokens  $i$  and  $j$ , let their inputs to an attention block be denoted as  $x_i$  and  $x_j$ ,  
 111 respectively. Then, their attention score  $\alpha_{ij}$  can be expressed as:

$$\begin{aligned}\hat{y}_i &= \text{LayerNorm}(y_i), \quad \hat{y}_j = \text{LayerNorm}(y_j), \\ \alpha_{ij} &= \text{Softmax}((\hat{y}_i W_q)(\hat{y}_j W_k)^T / \sqrt{d}),\end{aligned}\tag{7}$$



112 Based on the Observation 1,  $W_q W_k^T \approx \theta I$ , Therefore,

$$\begin{aligned}\alpha_{im} &= \text{Softmax}((\hat{y}_i W_q) (\hat{y}_j W_k)^T / \sqrt{d}) = \text{Softmax}(\langle \hat{y}_i W_q, \hat{y}_j W_k \rangle / \sqrt{d}) \\ &= \text{Softmax}(\hat{y}_i (W_q W_k^T) \hat{y}_j^T / \sqrt{d}) = \text{Softmax}(\theta \cdot \langle \hat{y}_i, \hat{y}_j \rangle / \sqrt{d})\end{aligned}\quad (8)$$

113 Furthermore, since LayerNorm preserves the direction of vectors by normalizing only their magnitude,  
114 the inner product between normalized outputs satisfies  $\langle \hat{y}_i, \hat{y}_j \rangle = \cos(\angle(y_i, y_j))$ ,

$$\alpha_{im} = \text{Softmax}(\theta \cdot \cos(\angle(y_i, y_j)) / \sqrt{d}) \propto \cos(\angle(y_i, y_j)) \quad (9)$$

115 Equation 9 indicates that the more aligned the hidden states of two input tokens are (i.e., the smaller  
116 the angle between them), the higher their attention score.

117 Therefore, in conjunction with the layer-wise angle concentration pattern discussed in the main  
118 text, we observe that inter-segment angular concentration reflects the model’s degree of attention to  
119 internal components of the problem, while intra-segment angular concentration captures the level  
120 of attention between the question and the system prompt—serving as an indicator of the model’s  
121 instruction-following capability.  $\square$

122 Theoretical Insight3. Presence of sink attention promotes angular concentration among hidden states.

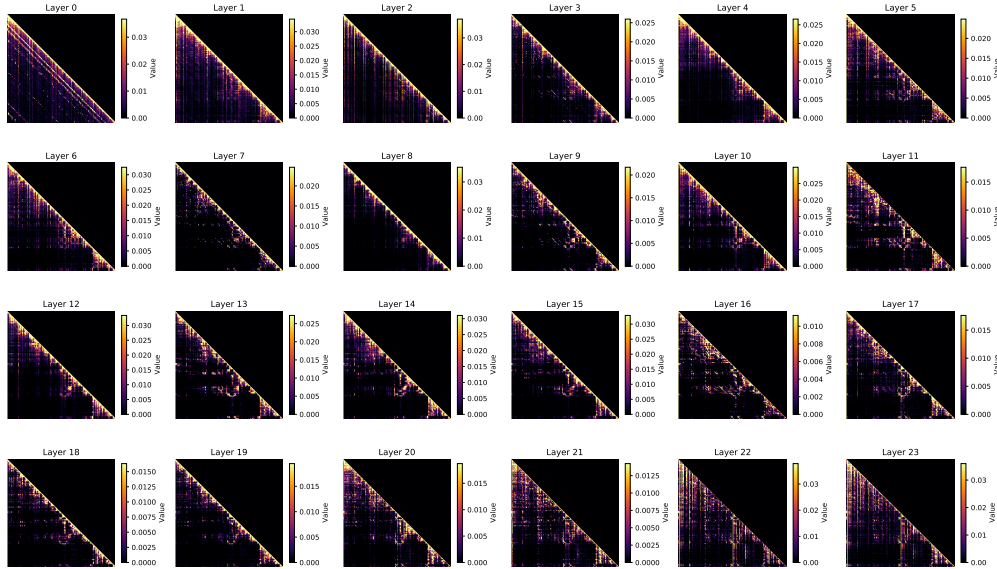


Figure 5: Visualization of attention score at different layers in Qwen2.5-0.5B-Instruct.

123 *Justification.* To understand why token angles exhibit Layer-wise angular concentration patterns,  
124 we start our analysis from the phenomenon of *sink attention*, which similarly shows segment-wise  
125 characteristics yet remains insufficiently understood. In LLMs, attention scores exhibit segment-wise  
126 tendencies, and, apart from self-attention, attention scores typically peak at the first token of each  
127 segment. This phenomenon is termed *sink attention*, as illustrated in Figure 5.

128 Given an input representation for an attention block in an LLM as  $\mathbf{y} \in \mathbb{R}^{m \times d}$ , where the  $i$ -th token  
129 is denoted as  $\mathbf{y}_i$ , the attention mechanism output is defined as  $\mathbf{O} = \alpha \mathbf{V}$ , where  $\alpha$  represents the  
130 attention scores, and  $\mathbf{V} = \text{LayerNorm}(\mathbf{y}) \mathbf{W}_v$  denotes the value vectors. Suppose the attention scores  
131 within a segment “sink” to the first token vector  $\mathbf{y}_i$ , then the outputs for the sink token  $\mathbf{y}_i$  and another  
132 token  $\mathbf{y}_k$  within the same segment can be approximated as:

$$\mathbf{O}_i \approx \alpha_{ii} \mathbf{V}_i, \quad \mathbf{O}_k \approx \alpha_{ik} \mathbf{V}_i + \alpha_{kk} \mathbf{V}_k,$$

133 where  $\alpha_{ik}$  denotes the attention score between tokens  $i$  and  $k$ . This approximation arises because  
134 the sink token’s self-attention score significantly surpasses its attention scores to other tokens, while  
135 other tokens primarily attend to themselves and the sink token.

136 Substituting these approximations, the angle between outputs  $\mathbf{O}_i$  and  $\mathbf{O}_k$  can be expressed as:

$$\cos(\angle(\mathbf{O}_i, \mathbf{O}_k)) = \frac{\alpha_{ik} \|\mathbf{V}_i\| + \alpha_{kk} \|\mathbf{V}_k\| \cos(\angle(\mathbf{V}_i, \mathbf{V}_k))}{\sqrt{\alpha_{ik}^2 \|\mathbf{V}_i\|^2 + \alpha_{kk}^2 \|\mathbf{V}_k\|^2 + 2\alpha_{ik}\alpha_{kk} \|\mathbf{V}_i\| \|\mathbf{V}_k\| \cos(\angle(\mathbf{y}_i, \mathbf{y}_k))}}$$

Furthermore, since  $\mathbf{W}_v$  is approximately an orthonormal matrix (detailed proof in the appendix) and thus preserves angles, we have  $\mathbf{W}_v \mathbf{W}_v^\top \approx \beta \mathbf{I}$ , where  $\beta$  is a constant. Combined with the fact that LayerNorm scales only magnitudes without altering angles, we get:

$$\cos(\angle(\mathbf{V}_i, \mathbf{V}_k)) = \cos(\angle(\mathbf{y}_i, \mathbf{y}_k)).$$

Substituting into the earlier expression, we derive:

$$\cos(\angle(\mathbf{O}_i, \mathbf{O}_k)) = \frac{\alpha_{ik} + \alpha_{kk} \cos(\angle(\mathbf{y}_i, \mathbf{y}_k))}{\sqrt{\beta \alpha_{ik}^2 + \alpha_{kk}^2 + 2\alpha_{ik}\alpha_{kk} \cos(\angle(\mathbf{y}_i, \mathbf{y}_k))}}$$

Squaring both sides and simplifying, we arrive at the condition:

$$\cos(\angle(\mathbf{O}_i, \mathbf{O}_k)) > \cos(\angle(\mathbf{y}_i, \mathbf{y}_k)) \quad \text{if} \quad \beta(\cos(\angle(\mathbf{y}_i, \mathbf{y}_k)))^2 < 1 \quad \text{and} \quad \beta < 1.$$

Since weight parameters in LLMs are typically constrained to values less than 1, both  $\beta < 1$  and  $\beta(\cos(\angle(\mathbf{y}_i, \mathbf{y}_k)))^2 < 1$  generally hold. Thus, Equation (7) demonstrates that sink attention inherently promotes angle concentration within segments. A detailed derivation is provided in the appendix.  $\square$

In Figure 5, we present the distribution of attention scores across different layers. In intermediate layers, sink tokens operate primarily within segments to enhance intra-segment angle concentration. In later layers, sink tokens across segments begin to interact, promoting inter-segment concentration. This indicates that, due to the influence of sink tokens, token angles are increasingly concentrated through the forward pass. The final layer’s angle concentration is particularly important as it reflects the culmination of this process and directly determines the model’s output.

### B.3 Neuron-Based Explanation of Data-wise Angle Concentration Patterns

In Section 2.3 of the main text, we demonstrate that the greater the number of tokens activating the same neuron, the more gradient components that neuron receive. In this section, we further show that the number of commonly activated neurons between tokens has a direct effect on the angle between their output hidden states at the current layer.

*Theoretical Insight 4. Within the FFN block, the extent of overlap in activated neurons between two tokens directly affects the angular relationship between their output hidden states.*

*Justification.* To analyze how the activation layer affects  $\cos(\angle(y_i, y_M))$ , we first decompose its computation formula. Suppose the activation output of the  $i$ -th token is  $A_i$ , and  $y_i = A_i W_d$ . Based on Observation 1,  $W_d$  is a scalar multiple of a unitary self-orthogonal matrix, applying the same rotation to any input while preserving the inner product and angle between any two input vectors. Thus, we can have:

$$\begin{aligned} \cos(\angle(y_i, y_M)) &= \langle A_i W_d, A_M W_d \rangle / (\|y_i\| \|y_M\|) \\ &= A_i (W_d W_d^T) A_M^T / (\|y_i\| \|y_M\|) \\ &= \eta \langle A_i, A_M \rangle / (\|y_i\| \|y_M\|), \end{aligned} \tag{10}$$

where  $\eta$  is a constant based on Observation 1. Furthermore, since  $W_d$  is an orthogonal matrix,

$$\begin{aligned} \|y_i\|^2 &= \|A_i W_d\|^2 = (A_i W_d)(A_i W_d)^T \\ &= A_i (W_d W_d^T) A_i^T = \eta A_i A_i^T = \eta \|A_i\|^2. \end{aligned} \tag{11}$$

which means  $\|y_i\| = \sqrt{\eta} \|A_i\|$ . Substituting this into Eq. 10 we can have

$$\begin{aligned} \cos(\angle(y_i, y_M)) &= \eta \langle A_i, A_M \rangle / (\eta \|A_i\| \|A_M\|) \\ &= \cos(\angle(A_i, A_M)) \end{aligned} \tag{12}$$

This shows that the orthogonal matrix  $W_d$  does not change the angles between the input vectors. Furthermore, based on Observation 2,  $\cos(\angle(A_i, A_M)) \propto |\Gamma(x_i) \cap \Gamma(x_M)|$ , we can get

$$\cos(\angle(y_i, y_M)) \propto |\Gamma(x_i) \cap \Gamma(x_M)|. \tag{13}$$

which is consistent with Insight 2.  $\square$

Eq. 13 shows that activation layers adjust token angles by controlling the intersections of their activated neurons. More shared activated neurons lead to smaller angles and greater mutual influence.

## C Visualization Results

In the main text, we presented layer-wise, epoch-wise, and data-wise patterns of angular concentration. In this section, we provide more comprehensive visualizations to further support our conclusions.

### C.1 Layer-wise Angle Concentration Patterns

In Fig. 6, 7 and 8, we present the layer-wise angular concentration patterns across all layers of the Qwen2.5-0.5B-Instruct model for tasks of easy, medium, and high difficulty, respectively. Notably, the model consistently demonstrates first intra-segment angle concentration and subsequently inter-segment angle concentration—regardless of problem difficulty. This consistency suggests that the observed pattern is a generalizable property of the model’s internal representation dynamics.

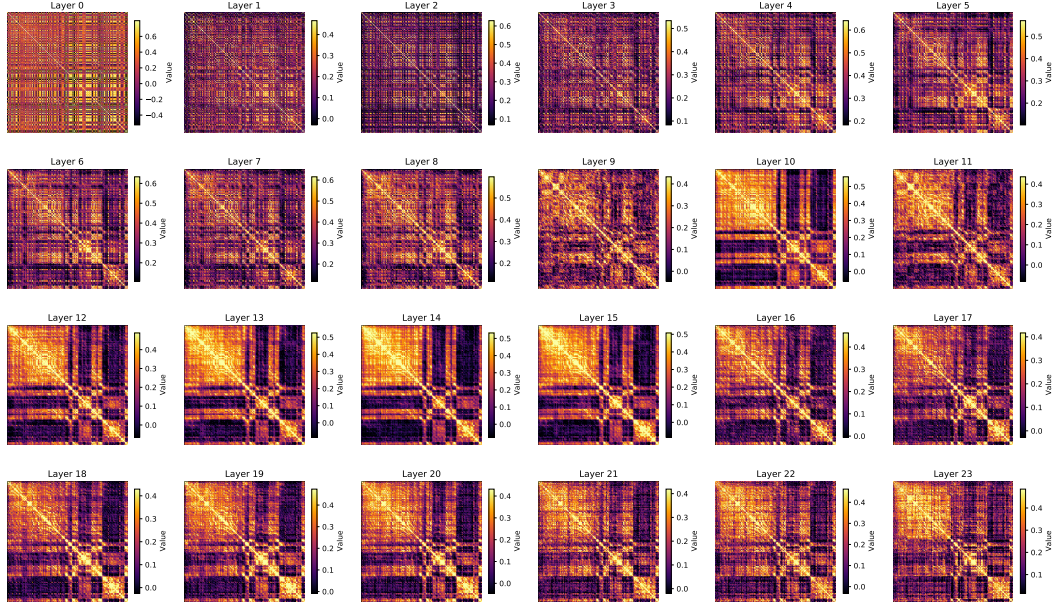


Figure 6: Visualization of Layer-wise Angle Concentration at different layers in Qwen2.5-0.5B-Instruct at easy sample (correct num = 10).

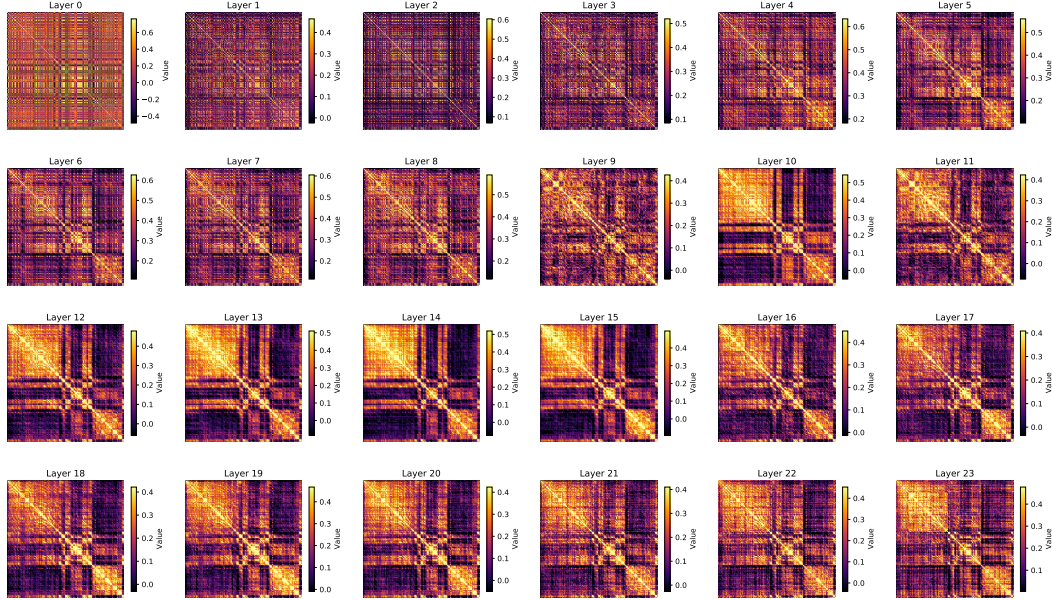


Figure 7: Visualization of Layer-wise Angle Concentration at different layers in Qwen2.5-0.5B-Instruct at medium difficulty sample (correct num = 5).



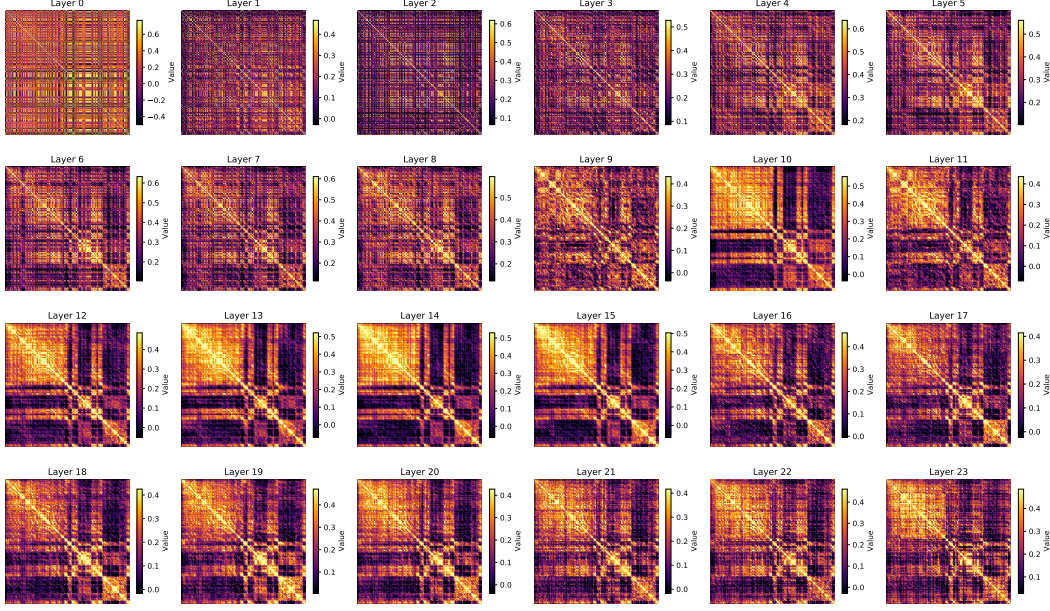


Figure 8: Visualization of Layer-wise Angle Concentration at different layers in Qwen2.5-0.5B-Instruct at difficult sample (correct num = 0).

## 180 C.2 Data-wise Angle Concentration Patterns

181 Fig. 9 provides a more fine-grained view of the data-wise angle-concentration patterns. Consistent  
 182 with the conclusions in the main text, it reveals a curriculum-like trend: the model learns samples  
 183 exhibiting high angular concentration earlier, followed by samples with lower angular concentration.

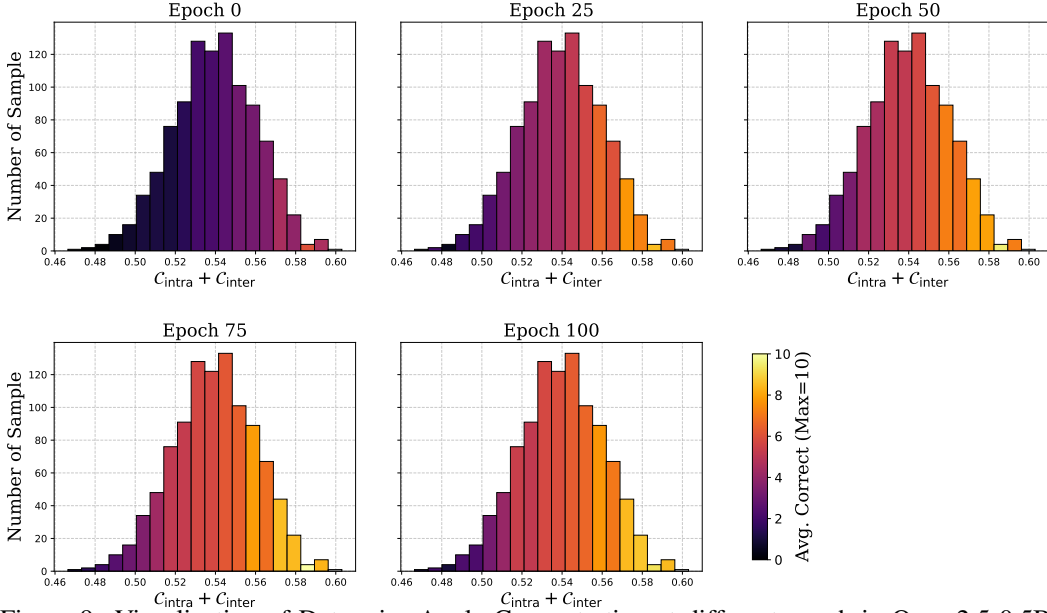


Figure 9: Visualization of Data-wise Angle Concentration at different epoch in Qwen2.5-0.5B-Instruct at difficult sample. The training setting is consistent with Figure 3 in the main text.

## 184 D GAIN-RL Algorithm

---

### Algorithm 1 GAIN-RL(GRPO) – Gradient-driven Angle-Informed Navigated RL Framework

---

```

1: Input: Training data  $D = \{d_1, d_2, \dots, d_N\}$ , model  $M$ , training step  $T$ , batch size  $n$ , accuracy
   sensitivity  $\alpha$ , target accuracy  $\beta$ , angle sensitivity  $\gamma$ , sampling variance  $\sigma$ .
2: Step1: Reordering Training Data based Angle Signal Before Training.
3: for  $i = 1$  to  $N$  do
4:   Perform prefilling on  $d_i$  using model  $M$ ;
5:   Extract and record the angle signal  $\mathcal{C}_M(d_i) = \mathcal{C}_{\text{intra}}^M(d_i) + \mathcal{C}_{\text{inter}}^M(d_i)$  from final-layer of  $M$ .
6: end for
7: Sort data  $D$  by angle signal  $\{\mathcal{C}_M(d_1), \mathcal{C}_M(d_2), \dots, \mathcal{C}_M(d_N)\}$  in descending order to obtain  $D_s$ .
8:
9: Step2: Training based on Dynamic Probabilistic Data Sampling.
10: Initialize a Gaussian probability distribution  $P_0 \sim \mathcal{N}(0, \sigma^2)$ .
11: for  $t = 1$  to  $T$  do
12:   Sample  $n$  samples from  $D_s$  according to probability  $P_t$ :  $d^{(t)} \sim \text{Sample}(D_s; P_t, n)$ ;
13:   for  $i = 1$  to  $n$  do
14:     Let the model of the  $t$ -th epoch  $M_t$  answer the sample  $d_i^{(t)}$ ;
15:     Collect the angle signal  $\mathcal{C}_{M_t}(d_i^{(t)})$  of the final layer of  $M_t$  and the accuracy  $\text{Acc}_{M_t}(d_i^{(t)})$ .
16:   end for
17:   Use accuracy to compute loss to update the model  $M_t$ .
18:   Compute mean accuracy  $\text{Acc}^{(t)}$  of all samples in  $d^{(t)}$ :  $\text{Acc}^{(t)} = \frac{1}{n} \sum_{i=1}^n \text{Acc}_{M_t}(d_i^{(t)})$ ;
19:   Compute mean angle signals  $\mathcal{C}^{(t)}$  of all samples in  $d^{(t)}$ :  $\mathcal{C}^{(t)} = \frac{1}{n} \sum_{i=1}^n \mathcal{C}_{M_t}(d_i^{(t)})$ ;
20:   Update sampling probability mean  $\mu_{t+1} = \mu_t + \frac{n}{2} \cdot \tanh\left(\alpha(\text{Acc}^{(t)} - \beta)\right) + \frac{n}{2} \cdot \tanh\left(\gamma \cdot \mathcal{C}^{(t)}\right)$ .
21: end for

```

---

## 185 E Experimental Setup

186 In this section, we describe our experimental setup in detail, covering the models, datasets, and  
 187 hyperparameters used.

### 188 E.1 Model and Dataset

189 To comprehensively evaluate the effectiveness of GAIN-RL, we conduct experiments across multiple  
 190 models and datasets. Specifically, we select models varying in size, including Qwen2.5-0.5b-  
 191 Instruct, Qwen2.5-Math-1.5B-Instruct, Qwen2.5-Math-7B-Instruct, Qwen2.5-Coder-3B-Instruct,  
 192 and LLaMA3.2-3B-Instruct. We primarily focus on two tasks: Math and Code. To evaluate the  
 193 training efficiency of GAIN-RL (Section 4.1 in the main text), we use the DeepScaleR [18] dataset for  
 194 mathematical task training and DeepCoder [19] for coding tasks training, each integrating problems  
 195 from diverse sources and covering a wide range of difficulty levels. For mathematical evaluations,  
 196 we employed six benchmark datasets of varying difficulty: GSM8K [20], MATH [21], AMC 23  
 197 [22], AIME 24 [23], OlympiadBench [24], and Minerva Math [25]. For coding evaluations, we  
 198 utilized three standard benchmark datasets: LivecodeBench (8/1/24–2/1/25) [26], Codeforces [27],  
 199 and Humaneval+ [28]. For other experiments (Section 4.2-Section 4.5), we train model on the  
 200 training dataset of single tasks including GSM8K, MATH and AMC 23 to facilitate more convenient  
 201 comparisons.

### 202 E.2 Training Configuration

203 We trained the models using the GRPO algorithm. The training was performed on a single node  
 204 equipped with 8 A100 GPUs. Each model was trained for about 200 steps using the veRL library.

205 To evaluate the training efficiency on GRPO-RL, the main training configuration for Qwen2.5-  
 206 Math-7B-Instruct is shown below. For Qwen2.5-Math-1.5B-Instruct and LLaMA3.2-3B-Instruct,  
 207 we set max\_response\_length=3000 to accommodate its shorter context window of 4096 to-  
 208 kens, while keeping all other parameters unchanged. For Qwen2.5-0.5B-Instruct and single task

209 training, we set `max_prompt_length=max_response_length=512`, while keeping other param-  
 210 eters unchanged. For Qwen/Qwen2.5-Coder-3B-Instruct, we set `max_prompt_length=2048`,  
 211 `max_response_length=16384`, `train_batch_size=512` and `ppo_mini_batch_size=64` due  
 212 to its higher single sample memory usage.

```

213 python3 -m verl.trainer.main_ppo \
214     algorithm.adv_estimator=grpo \
215     data.train_files="$train_files" \
216     data.val_files="$test_files" \
217     data.train_batch_size=1024 \
218     data.max_prompt_length=1024 \
219     data.max_response_length=8192 \
220     actor_rollout_ref.model.path=Qwen/Qwen2.5-Math-7B-Instruct \
221     actor_rollout_ref.actor.optim.lr=1e-6 \
222     actor_rollout_ref.model.use_remove_padding=True \
223     actor_rollout_ref.actor.ppo_mini_batch_size=256 \
224     actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=16 \
225     actor_rollout_ref.actor.use_dynamic_bsz=True \
226     actor_rollout_ref.actor.ppo_max_token_len_per_gpu=8000 \
227     actor_rollout_ref.actor.use_kl_loss=True \
228     actor_rollout_ref.actor.kl_loss_coef=0.001 \
229     actor_rollout_ref.actor.kl_loss_type=low_var_kl \
230     actor_rollout_ref.actor.entropy_coef=0 \
231     actor_rollout_ref.model.enable_gradient_checkpointing=True \
232     actor_rollout_ref.actor.fsdps_config.param_offload=False \
233     actor_rollout_ref.actor.fsdps_config.optimizer_offload=False \
234     actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
235     actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=16 \
236     actor_rollout_ref.rollout.name=vllm \
237     actor_rollout_ref.rollout.gpu_memory_utilization=0.6 \
238     actor_rollout_ref.rollout.n=8 \
239     actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=16 \
240     actor_rollout_ref.ref.fsdps_config.param_offload=True \
241     algorithm.use_kl_in_reward=False \
242     trainer.critic_warmup=0 \
243     trainer.logger=['console', 'wandb'] \
244     trainer.project_name='verl_grpo_example_gsm8k_math' \
245     trainer.experiment_name='qwen2_MATH_7b-Instruct_function_rm' \
246     trainer.n_gpus_per_node=8 \
247     trainer.nnodes=1 \
248     trainer.save_freq=20 \
249     trainer.test_freq=5 \
250     trainer.total_epochs=200 $@

```

## 251 F Additional Experimental Results

252 In this section, we present additional experiments to further validate the effectiveness of GAIN-RL.

### 253 F.1 Performance of Weighted Signals

254 In the main text, we employed an unweighted angular signal of the form:  $\mathcal{C}_M(d_i) = \mathcal{C}_{\text{intra}}^M(d_i) +$   
 255  $\mathcal{C}_{\text{inter}}^M(d_i)$ . Here, we explore a weighted variant of the signal:  $\mathcal{C}_M(d_i) = \mathcal{C}_{\text{intra}}^M(d_i) + c \cdot \mathcal{C}_{\text{inter}}^M(d_i)$ ,  
 256 where  $c$  is a constant weight.

257 As shown in Tab. 1, we investigate the final performance of the Qwen2.5-0.5B-Instruct model after  
 258 200 training epochs on the Math dataset, using different values of  $c$  in the weighted signal. The  
 259 results show that setting  $c = 4.0$  yields the best performance for this model, suggesting that carefully  
 260 designed angle-based signals can further improve training effectiveness.



However, selecting the optimal weighting coefficient  $c$  requires extensive empirical tuning. To ensure scalability and practical applicability, we adopt the unweighted version of the signal in this work and leave signal optimization for future research. Our goal is to demonstrate that *even without finely tuned weighting*, GAIN-RL is still capable of accelerating both training and data efficiency—highlighting the strong potential of model-signal-based RLHF strategies.

Table 1: Model performance at epoch 200 under different weighting coefficients  $c$ .

$c$	0.25	0.5	1.0	2.0	4.0	8.0
Accuracy	36.20	37.20	37.40	38.00	38.40	38.20

## F.2 Model Performance on Single Task

Fig. 9 in the main text illustrates the training dynamics of Qwen-2.5-0.5B-Instruct on three single-task datasets. For a more detailed comparison, Tab. 2 reports the final performance at epoch 200 and the corresponding training speedup across different training sets and methods. Notably, on the GSM8K dataset, GAIN-RL outperforms the original GRPO by 4.72% in final accuracy and achieves a  $3.3\times$  improvement in training speed.

These results demonstrate that GAIN-RL can effectively distinguish between samples of varying learnability even in the single-task setting, highlighting its efficiency and general applicability.

Table 2: Fine-tuning performance on single tasks. Models are trained on the training sets and evaluated on their validation sets. ADARFT is excluded on GSM8K due to missing difficulty coefficients.

	Prepare		GSM8K			Math			AMC		
	Metric	Time	ACC@ 200Epo	Epo@ 200Acc	Speed Up	ACC@ 200Epo	Epo@ 200Acc	Speed Up	ACC@ 200Epo	Epo@ 200Acc	Speed Up
GRPO	-	-	48.43	200	$1\times$	34.80	200	$1\times$	9.64	200	$1\times$
ADARFT(GRPO)	Difficulty	> 1 day	-	-	-	35.80	150	$1.33\times$	9.64	160	$1.25\times$
GAIN-RL(GRPO)	Angle	< 10 min	53.15	60	$3.33\times$	37.40	80	$2.50\times$	12.04	100	$2.00\times$

## G Discussion and Future Work

In Section 3, we demonstrate that angles between token hidden states fundamentally mirror and influence both the information propagation during inference and the learning dynamics throughout model training. The proposed angle-based signals can, in fact, be generalized beyond RFT to enhance model-centric effectiveness in various other domains. For instance, during pre-training, monitoring angle signals could enable real-time evaluation of a model’s learning capacity across different domains, thus allowing adjustments to training data to improve stability and final performance. Furthermore, during inference, tracking changes in angle concentration between layers could provide insights into the model’s comprehension of inputs and indicate whether additional test-time adjustments are necessary to boost output accuracy. In future work, we plan to further investigate how this signal can be leveraged across multiple domains to achieve comprehensive, model-centric optimizations.

## References

- [1] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [2] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.
- [3] Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

- [4] Judy Hanwen Shen, Archit Sharma, and Jun Qin. Towards data-centric rlhf: Simple metrics for preference dataset comparison. In *NeurIPS*, 2024. URL <https://arxiv.org/abs/2409.09603>.
- [5] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.
- [6] Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt J Kusner. No train no gain: Revisiting efficient training algorithms for transformer-based language models. *Advances in Neural Information Processing Systems*, 36:25793–25818, 2023.
- [7] Dante Everaert and Christopher Potts. Gio: Gradient information optimization for training dataset selection. *arXiv preprint arXiv:2306.11670*, 2023.
- [8] Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.
- [9] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.
- [10] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.
- [11] Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Active preference optimization for sample efficient rlhf. *arXiv preprint arXiv:2402.10500*, 2024.
- [12] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
- [13] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- [14] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- [15] Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025.
- [16] Shuai Li, Kui Jia, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019.
- [17] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? *Advances in Neural Information Processing Systems*, 31, 2018.
- [18] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.
- [19] Michael Luo, Sijun Tan, Roy Huang, Xiaoxiang Shi, Rachel Xin, Colin Cai, Ameen Patel, Alpay Ariyak, Qingyang Wu, Ce Zhang, et al. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025.
- [20] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

- 340 [21] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn  
341 Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset.  
342 *arXiv preprint arXiv:2103.03874*, 2021.
- 343 [22] *Proceedings of the ACM Web Conference 2023*, Austin, TX, USA, 2023. Association for Com-  
344 puting Machinery. URL <https://dl.acm.org/doi/proceedings/10.1145/3543507>.
- 345 [23] 2024 american invitational mathematics examination (aime). [https://www.maa.org/](https://www.maa.org/math-competitions)  
346 [math-competitions](https://www.maa.org/math-competitions), 2024. Accessed: 2025-05-14.
- 347 [24] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu,  
348 Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for  
349 promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint*  
350 *arXiv:2402.14008*, 2024.
- 351 [25] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay  
352 Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving  
353 quantitative reasoning problems with language models. *Advances in Neural Information*  
354 *Processing Systems*, 35:3843–3857, 2022.
- 355 [26] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Ar-  
356 mando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination  
357 free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- 358 [27] Mikhail Mirzayanov. Open codeforces rating system. [https://codeforces.com/blog/](https://codeforces.com/blog/entry/20762)  
359 [entry/20762](https://codeforces.com/blog/entry/20762), 2015. Accessed: 2025-05-14.
- 360 [28] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared  
361 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large  
362 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.