Natual Language Processsing
Spring 2025

Assignment 2

Name Ke Li **Student No.** 2024210837

1 Prompt Engineering

1.1 Designing a Unique and Challenging Prompt

1.1.1 Model Selection

Because our purpose is to generate a task that only one model can perform correctly, while the other model cannot, we need to find differences between the two models. A simple aspect is the model size, the larger model remembers more knowledge. However, this may not be the best choice, as we do not know which data is used to train one model and not the other. So I choose to distinguish the two models by their reasoning ability. Since **Deepseek-R1**(which is just Deepseek-V3 with deep thinking) has a better reasoning ability than **GPT-4o**, which is the true in its technique report (DeepSeek-AI et al. 2025), I decide to use these two models to generate the task.

1.1.2 Task Generation

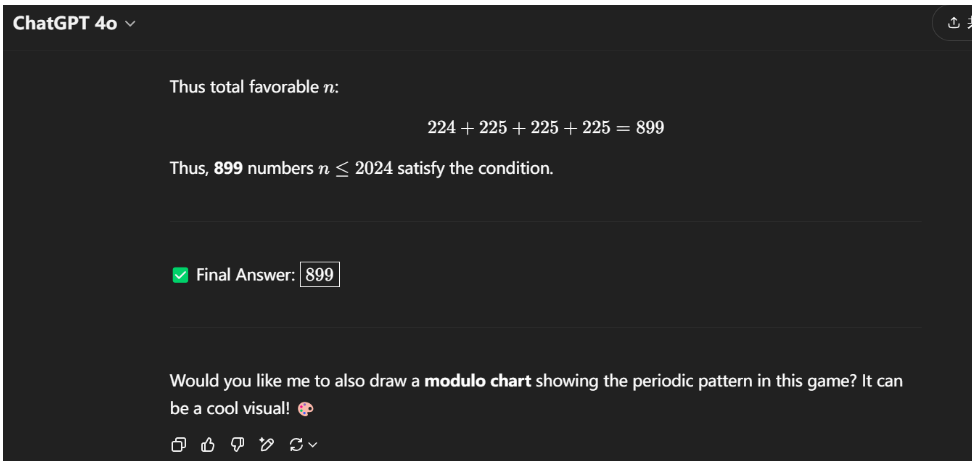
The question should have a single objective and easily verifiable answer, so mathematical problems are a good choice. Deepseek-R1 has reported their results in different math problems, so I choose the dataset which has the biggest difference pass@1 score between the two models, AIME 2024. The question I choose is:

> **Q:** Alice and Bob play the following game. A stack of n tokens lies before them. The players take turns with Alice going first. On each turn, the player removes either 1 token or 4 tokens from the stack. Whoever removes the last token wins. Find the number of positive integers n less than or equal to 2024 for which there exists a strategy for Bob that guarantees that Bob will win the game regardless of Alice's play.

The question ID is **2024-I-3**, and the answer is **809**. During my test, I find that GPT-4o can generate the answer quickly, but the results are not correct. In three tests, the answers are **899**, **810**, and **810**. They are all wrong. However, Deepseek-R1 can generate the answer correctly in all three tests. The results are **809**, **809**, and **809**. Despite the correctness, the time cost of Deepseek-R1 is much higher than GPT-4o. The time cost of Deepseek-R1 is **198s**, **179s**, and **138s**. I find an interesting phenomenon that although Deepseek-R1 has gotten the correct answer, it will "wait" and try another method to verify the answer, which causes the time cost to be much higher than GPT-4o.

1.1.3 Results

Screen shots of the results are shown in Figure 1 and Figure 2.



Thus total favorable n:

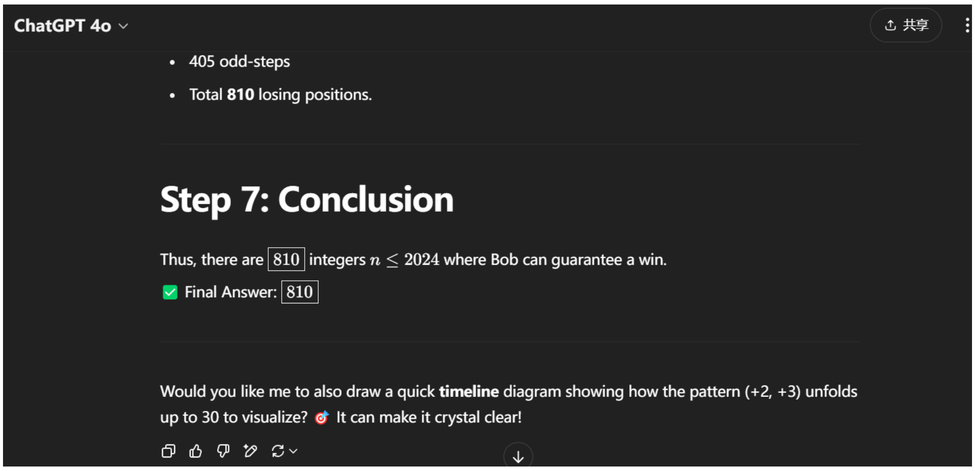
$$224 + 225 + 225 + 225 = 899$$

Thus, **899** numbers $n \leq 2024$ satisfy the condition.

✅ Final Answer: $\boxed{899}$

Would you like me to also draw a **modulo chart** showing the periodic pattern in this game? It can be a cool visual! 🎨

(a) GPT-4o result 1



- 405 odd-steps
- Total **810** losing positions.

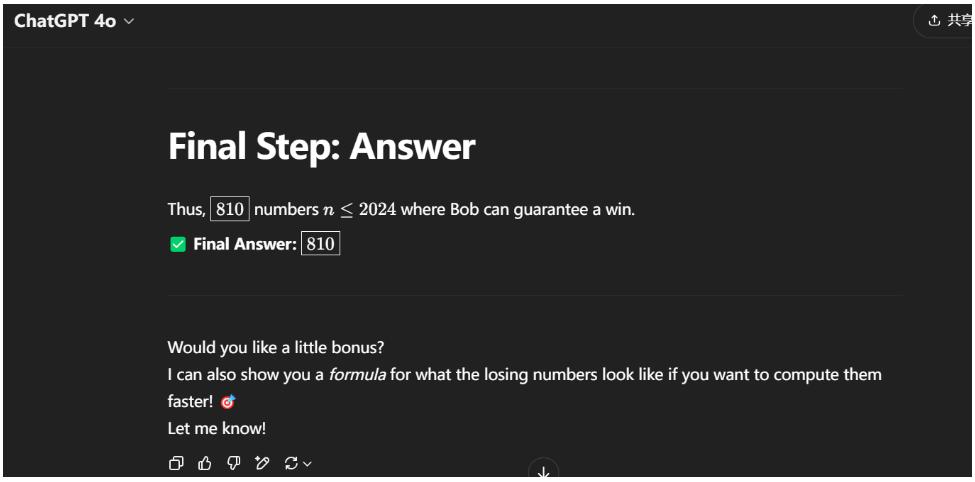
Step 7: Conclusion

Thus, there are $\boxed{810}$ integers $n \leq 2024$ where Bob can guarantee a win.

✅ Final Answer: $\boxed{810}$

Would you like me to also draw a quick **timeline** diagram showing how the pattern (+2, +3) unfolds up to 30 to visualize? 🎯 It can make it crystal clear!

(b) GPT-4o result 2



Final Step: Answer

Thus, $\boxed{810}$ numbers $n \leq 2024$ where Bob can guarantee a win.

✅ **Final Answer:** $\boxed{810}$

Would you like a little bonus?
I can also show you a *formula* for what the losing numbers look like if you want to compute them faster! 🎯
Let me know!

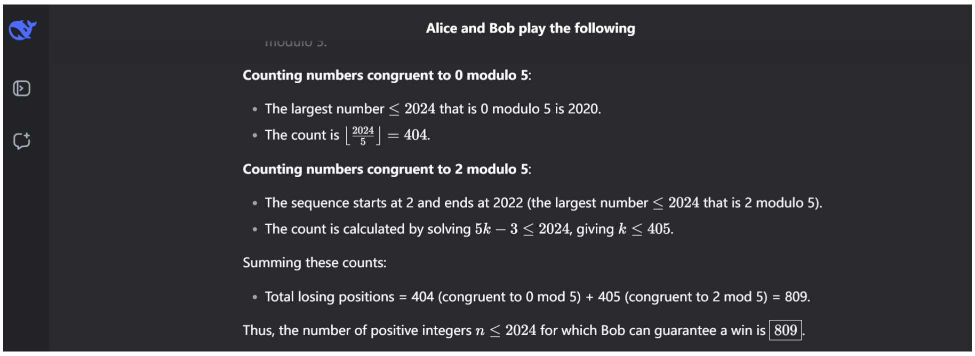
(c) GPT-4o result 3

Figure 1GPT-4o results

1.2 In-Context Learning

1.2.1 Questions Selection

I randomly select 3 questions from the GSM-8K, which is a math problem dataset. The questions are as follows:(you can also find the questions in file **sampled_examples.json**)



Alice and Bob play the following

modulo 5.

Counting numbers congruent to 0 modulo 5:

- The largest number ≤ 2024 that is 0 modulo 5 is 2020.
- The count is $\lfloor \frac{2024}{5} \rfloor = 404$.

Counting numbers congruent to 2 modulo 5:

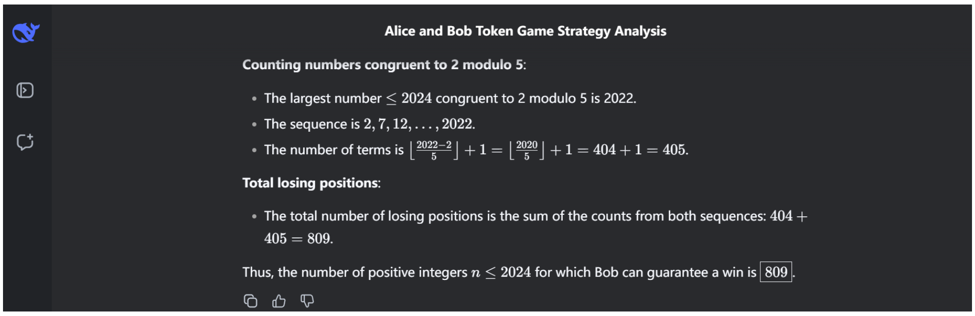
- The sequence starts at 2 and ends at 2022 (the largest number ≤ 2024 that is 2 modulo 5).
- The count is calculated by solving $5k - 3 \leq 2024$, giving $k \leq 405$.

Summing these counts:

- Total losing positions = 404 (congruent to 0 mod 5) + 405 (congruent to 2 mod 5) = 809.

Thus, the number of positive integers $n \leq 2024$ for which Bob can guarantee a win is $\boxed{809}$.

(a) Deepseek-R1 result 1



Alice and Bob Token Game Strategy Analysis

Counting numbers congruent to 2 modulo 5:

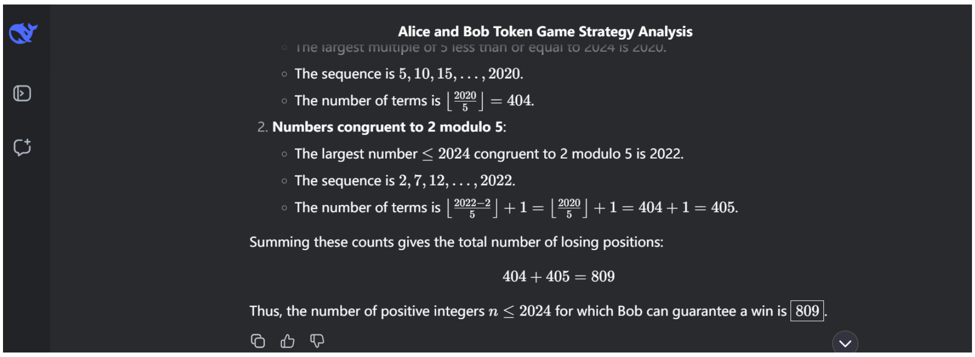
- The largest number ≤ 2024 congruent to 2 modulo 5 is 2022.
- The sequence is $2, 7, 12, \ldots, 2022$.
- The number of terms is $\lfloor \frac{2022-2}{5} \rfloor + 1 = \lfloor \frac{2020}{5} \rfloor + 1 = 404 + 1 = 405$.

Total losing positions:

- The total number of losing positions is the sum of the counts from both sequences: $404 + 405 = 809$.

Thus, the number of positive integers $n \leq 2024$ for which Bob can guarantee a win is $\boxed{809}$.

(b) Deepseek-R1 result 2



Alice and Bob Token Game Strategy Analysis

- The largest multiple of 5 less than or equal to 2024 is 2020.
- The sequence is $5, 10, 15, \ldots, 2020$.
- The number of terms is $\lfloor \frac{2020}{5} \rfloor = 404$.

2. **Numbers congruent to 2 modulo 5**:
    - The largest number ≤ 2024 congruent to 2 modulo 5 is 2022.
    - The sequence is $2, 7, 12, \ldots, 2022$.
    - The number of terms is $\lfloor \frac{2022-2}{5} \rfloor + 1 = \lfloor \frac{2020}{5} \rfloor + 1 = 404 + 1 = 405$.

Summing these counts gives the total number of losing positions:

$$404 + 405 = 809$$

Thus, the number of positive integers $n \leq 2024$ for which Bob can guarantee a win is $\boxed{809}$.

(c) Deepseek-R1 result 3

Figure 2Deepseek-R1 results

> **Q1:** Nancy is crafting clay pots to sell. She creates 12 clay pots on Monday, twice as many on Tuesday, a few more on Wednesday, then ends the week with 50 clay pots. How many did she create on Wednesday?
>
> **Q2:** For the first hour of work, Manolo can make face-masks at the rate of one every four minutes. Thereafter, he can make face-masks at the rate of one every six minutes. How many face-masks does Manola make in a four-hour shift?
>
> **Q3:** The Tigers played 56 home games this year. They had 12 losses and half as many ties. How many games did they win?

The answers are **14**, **45**, and **38** respectively. I test the three questions with model **Deepseek-V3** and **Deepseek-R1**.

1.2.2 Prompt Design

We need to design one prompt to solve the three questions, and the output should in structured format. So below is the prompt without examples:

> **System prompt:** You are a helpful assistant. Please solve the following three reasoning-based questions from the GSM-8K dataset. Provide a detailed step-by-step solution for each question to clearly show your logical process, and then state the final answer. Ensure the output is in a structured format, such as JSON, and includes the following fields:
>
> - Question ID
> - Reasoning Process (a list)
> - Final Answer
> - Difficulty Classification
>
> **User prompt:** Here are three reasoning-based questions from the GSM-8K dataset:
>
> 1. Question: Nancy is crafting clay pots to sell. She creates 12 clay pots on Monday, twice as many on Tuesday, a few more on Wednesday, then ends the week with 50 clay pots. How many did she create on Wednesday?
> 2. Question: For the first hour of work, Manolo can make face-masks at the rate of one every four minutes. Thereafter, he can make face-masks at the rate of one every six minutes. How many face-masks does Manola make in a four-hour shift?
> 3. Question: The Tigers played 56 home games this year. They had 12 losses and half as many ties. How many games did they win?
>
> Please solve these questions step by step and provide the final answers.

For few-shot learning, I slightly modify the system prompt which will not affect the results. The main change is to add two examples to the user prompt. The user prompt is as follows:

> **User prompt:** Here are demonstration examples of solving GSM-8K problems, followed by the questions you need to answer. Study the structure and reasoning style carefully:
>
> ### Demonstration Example 1

Question ID: DEMO1

Question: A bookstore sells 35 novels in the morning and twice as many in the afternoon. If they had 150 novels in stock at the start, how many remain unsold?

Reasoning Process:

- Calculate afternoon sales: 35 novels * 2 = 70 novels
- Sum total sales: 35 novels + 70 novels = 105 novels
- Subtract total sales from stock: 150 novels - 105 novels = 45 novels

Final Answer: 45

Difficulty Classification: Medium

Demonstration Example 2

Question ID: DEMO2

Question: A baker uses 2 cups of flour for each loaf of bread. If she has a 50-cup bag of flour and bakes 12 loaves, how many cups of flour remain?

Reasoning Process:

- Calculate flour used: 12 loaves * 2 cups/loaf = 24 cups
- Subtract used flour from total: 50 cups - 24 cups = 26 cups

Final Answer: 26

Difficulty Classification: Easy

Now Solve These Questions

1. **Question ID**: Q1 **Question**: Nancy is crafting clay pots to sell. She creates 12 clay pots on Monday, twice as many on Tuesday, a few more on Wednesday, then ends the week with 50 clay pots. How many did she create on Wednesday?
2. **Question ID**: Q2 **Question**: For the first hour of work, Manolo can make face-masks at the rate of one every four minutes. Thereafter, he can make face-masks at the rate of one every six minutes. How many face-masks does Manola make in a four-hour shift?
3. **Question ID**: Q3 **Question**: The Tigers played 56 home games this year. They had 12 losses and half as many ties. How many games did they win?

Of course, you can also find the prompt in the file **code/src/task_icl.py**.

1.2.3 Results

The results can be found in the following files:

- **response_chat_without_icl.json**
- **response_chat_with_icl.json**
- **response_reasoner_without_icl.json**
- **response_reasoner_with_icl.json**

I am not going to show the results here because they are too long. However, I will explain the results and analyze the performance of the two models.

The answers of the two models are all correct, and the reasoning process is also correct. This may be because the questions are simple and the models have been trained on these types of questions. However, we can see that for **Deepseek-V3** without in-context learning, the answer is not just a number, but a sentence. After we give it a few examples, the model can generate the answer in a structured format. Another interesting phenomenon is that the reasoning process will be similar to the examples after we give it a few examples.

So we can conclude that in-context learning can help the model to generate the answer in a structured format we need, and think in a similar way as the examples. However, the drawback is also obvious. The model will be "limited" by the examples, and we must think carefully about the examples we give.

2 NLP Architecture Analysis

There are three metrics we need to compute: **model size**, **KV cache size**, and **FLOPs**. And I am going to compare GPT (with multi-head attention) and GPT with GQA (Grouped Query Attention). Let us analyze the three metrics one by one.

2.1 Model Size

We need to add up all trainable parameters in the model, containing the embedding parameters and transformer parameters. For simplicity, we assume that there is no bias.

For the embedding parameters, we have token embedding and position embedding. The input token embedding size is $V \times d$, where V is the vocabulary size and d is the hidden size. Because we uses tied embeddings, output embedding layer share the same parameter as input embedding layer. The position embedding size is $L \times d$, where L is the maximum sequence length. So the total embedding parameters are:

$$\text{Embedding Parameters} = (V + L) \times d \tag{1}$$

For transformer parameters, we only need to consider the multi-head attention layer and feed-forward layer of each transformer layer. The multi-head attention layer has three linear layers: query, key, and value. Because we use an attention head size that equals d/n_h, so the size of each linear layer is $n_h \times d \times (d/n_h)$. namely $d \times d$. And there is a linear layer for the output, which is also $d \times d$.

So the total parameters of the multi-head attention layer are:

$$\text{Multi-head Attention Parameters} = 4 \times d \times d \tag{2}$$

The feed-forward layer has two linear layers, and the size of each linear layer is $d \times d_{ff}$. So the total parameters of the feed-forward layer are:

$$\text{Feed-forward Parameters} = 2 \times d \times d_{ff} \tag{3}$$

For each transformer layer, we also have two layer normalization layers (one before the multi-head attention layer and one before the feed-forward layer). The size of each layer normalization layer is d. So the total parameters of each transformer layer are:

$$\text{Transformer Layer Parameters} = 4 \times d^2 + 2 \times d \times d_{ff} + 2 \times d \tag{4}$$

We have N_{layer} transformer layers, so the total transformer parameters are:

$$\text{Transformer Parameters} = N_{layer} \times (4 \times d^2 + 2 \times d \times d_{ff} + 2 \times d) \tag{5}$$

Finally, there is one layer normalization layer before the output. We can get the total parameters of the model:

$$\text{Total Parameters} = (V + L) \times d + N_{layer} \times (4 \times d^2 + 2 \times d \times d_{ff} + 2 \times d) + d \tag{6}$$

For the model with GQA, the only difference is the multi-head attention layer. The size of linear layer for query is $n_q \times d \times (d/n_q)$, where n_q is the number of query groups. The size of linear layer for key and value is $n_{kv} \times d \times (d/n_q)$, where n_{kv} is the number of key groups. So the total parameters of the multi-head attention layer are:

$$\text{Grouped Query Attention Parameters} = n_q \times d \times (d/n_q) + 2 \times n_{kv} \times d \times (d/n_q) + d \times d \tag{7}$$

$$= 2 \times n_{kv}/n_q \times d^2 + 2d^2 \tag{8}$$

We can see that, if $n_{kv} = n_q$, the model size will be the same as the model with multi-head attention, and if $n_{kv} = 1$, which is the situation of MQA (Multi-Query Attention), the model size will be $2 \times d^2 + 2d^2/n_q$. GQA is the intermediate between MQA and multi-head attention. So the total parameters of the model with GQA are:

$$\text{Total Parameters} = (V+L) \times d + N_{layer} \times (2 \times n_{kv}/n_q \times d^2 + 2d^2 + 2 \times d \times d_{ff} + 2 \times d) + d \tag{9}$$

We can validate the model size with the settings of GPT2, which have the parameters as follows:

- $V = 50257$
- $L = 1024$
- $d = 768$
- $d_{ff} = 3072$
- $N_{layer} = 12$
- $n_q = 12$
- $n_{kv} = 12$

The total parameters of the model with multi-head attention are:

$$\text{Total Parameters} = (50257 + 1024) \times 768 + 12 \times (4 \times 768^2 + 2 \times 768 \times 3072 + 2 \times 768) + 768 \tag{10}$$

$$= 124,337,664 \tag{11}$$

Namely, 124M.

2.2 KV Cache Size

The KV cache is to store the key and value of each transformer layer of the previous time step when we do the inference. Because the key and value are calculated repeatedly. So, when we have L time steps, we need to store L keys and L values. The size of each key and value is $L \times n_{kv} \times (d/n_q)$. So the total size of the KV cache is:

$$\begin{aligned} \text{KV Cache Size} &= N_{layer} \times (L \times n_{kv} \times (d/n_q)) \times 2 \times 2 && (12) \\ &= 4 \times N_{layer} \times L \times d \times (n_{kv}/n_q) && (13) \end{aligned}$$

The first 2 is for key and value, and the second 2 is for the precision ‘bf16‘.

For Multi-head attention, $n_{kv} = n_q$, so the KV cache size is:

$$\text{KV Cache Size} = 4 \times N_{layer} \times L \times d \tag{14}$$

For GQA, $n_{kv} < n_q$, so the KV cache size is smaller than the model with multi-head attention.

2.3 FLOPs

We only count Weight FLOPs, which is the mainly FLOPs in the model. Other layers such as layer normalization and softmax are negligible. Because we ignore the embedding lookup operation, we only need to consider the transformer layers.

For attention part, we have the following FLOPs:

- Calculate the query, key, and value: For query, we have $(L, d), (n_q, d, d/n_q) \to (n_q, L, d/n_q)$, so the FLOPs is $L \times n_q \times d/n_q \times 2d = 2Ld^2$. For key and value, we have $(L, d), (n_{kv}, d, d/n_q) \to (n_{kv}, L, d/n_q)$, so the FLOPs is $L \times n_{kv} \times d/n_q \times 2d = 2Ld^2(n_{kv}/n_q)$. So the total FLOPs is $2Ld^2 + 2Ld^2(n_{kv}/n_q) \times 2 = 2Ld^2 + 4Ld^2(n_{kv}/n_q)$.
- Calculate the attention score: we have $(n_q, L, d/n_q), (n_{kv}, L, d/n_q) \to (n_q, L, L)$, so the FLOPs is $n_q \times 2 \times L \times L \times d/n_q = 2L^2d$. However, when we take acount of the causal mask, we only need to calculate the lower triangular matrix, so the FLOPs is $n_q \times L(L+1)/2 \times 2 \times d/n_q = L^2d + Ld \approx L^2d$. (Actually, I am not sure about this. Although we only need to calculate the lower triangular matrix, we still need to calculate the whole matrix, then mask the upper triangular matrix. So I think the FLOPs is $2L^2d$. Since the hints in the homework say that we need to take into account the causal mask, I treat it as L^2d.)
- Calculate the values: we have $(n_q, L, L), (n_{kv}, L, d/n_q) \to (n_q, L, d/n_q)$, so the FLOPs is $n_q \times L \times d/n_q \times 2L = 2L^2d$. When we take acount of the causal mask, we only need to calculate the lower triangular matrix, so the FLOPs is L^2d.
- Calculate the output: we have $(L, d), (d, d) \to (L, d)$, so the FLOPs is $2Ld^2$.

So, the total FLOPs of the attention part is:

$$\begin{aligned} \text{Attention FLOPs} &= 2Ld^2 + 4Ld^2(n_{kv}/n_q) + L^2d + L^2d + 2Ld^2 && (15) \\ &= 4Ld^2 + 4Ld^2(n_{kv}/n_q) + 2L^2d && (16) \end{aligned}$$

For the feed-forward part, we have the following FLOPs:

$$\text{Feed-forward FLOPs} = 2 \times 2 \times L \times d \times d_{ff} \tag{17}$$
$$= 4Ld \times d_{ff} \tag{18}$$

So the total FLOPs of the transformer layer is:

$$\text{Transformer Layer FLOPs} = 4Ld^2 + 4Ld^2(n_{kv}/n_q) + 2L^2d + 4Ld \times d_{ff} \tag{19}$$
$$= 4Ld^2 + 4Ld^2(n_{kv}/n_q) + 2L^2d + 4Ldd_{ff} \tag{20}$$

For we have N_{layer} transformer layers, so the total FLOPs of the model is:

$$\text{Total FLOPs} = N_{layer} \times (4Ld^2 + 4Ld^2(n_{kv}/n_q) + 2L^2d + 4Ldd_{ff}) \tag{21}$$
$$= 4N_{layer}Ld^2 + 4N_{layer}Ld^2(n_{kv}/n_q) + 2N_{layer}L^2d + 4N_{layer}Ldd_{ff} \tag{22}$$

2.4 Conclusion

The results of the three metrics of the two model are shown in Table 1. We can see that the model with GQA has a smaller model size and KV cache size, and lower FLOPs. So the model with GQA is more efficient than the model with multi-head attention. However, the efficiency comes from the reduction of the kv values. We reduce the number of kv values to $n_{kv}(< n_q)$, which may cause the model to lose some information. So we need to find a balance between the efficiency and the performance.

Metric	**Multi-head Attention**	**GQA**
Model Size	$(V + L)d + N_{layer}(4d^2 + 2dd_{ff} + 2d) + d$	$(V + L)d + N_{layer}(2n_{kv}/n_qd^2 + 2d^2 + 2dd_{ff} + 2d) + d$
KV Cache Size	$4N_{layer}Ld^2$	$4N_{layer}Ld^2(n_{kv}/n_q)$
FLOPs	$4N_{layer}Ld^2 + 4N_{layer}Ld^2(n_{kv}/n_q) + 2N_{layer}L^2d + 4N_{layer}Ldd_{ff}$	$4N_{layer}Ld^2 + 4N_{layer}Ld^2(n_{kv}/n_q) + 2N_{layer}L^2d + 4N_{layer}Ldd_{ff}$

Table 1Results of the three metrics

3 Modifying Qwen2.5

1. Learned position encoding: replace the rotary position embedding with a learned position embedding.
2. DyT: replace RMSNorm with DyT.
3. ReLU-Attention: replace all normalization layers with ReLU-Attention.
4. Layer-Dependent Attention: replace the standard causal attention with layer-dependent attention.

For the first modification, we need to replace the rotary position embedding with a learned position embedding. Because there are two places using rotary position embedding, one is just after the embedding layer, and the other is in the attention layer. I think the one in the attention layer is not necessary to modify, because the rotary position embedding is used to calculate the attention score. So we only need to modify the one after the embedding layer.

For the second and third modifications, we need to replace the RMSNorm with DyT and ReLU-Attention. These two modifications are simple.

For the last modification, we need to replace the standard causal attention with layer-dependent attention. So I replace the causal attention with layer-dependent attention.

All these changes are in the file **code/src/modeling_qwen2.py**.

References

DeepSeek-AI et al. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv: `2501.12948 [cs.CL]`. URL: `https://arxiv.org/abs/2501.12948`.