author - Ataka jeong

1. Introduction

How could the DeepSeek-V3 model achieve incredible performance and economical training as an open source model? In this paper review, we will explore the various features that were invented and applied to build the DeepSeek-V3 model. The way the paper presents the model may seem complicated to people who are unfamiliar with the new conceptualization invented by DeepSeek. However, its core principle still resembles that of the standard Transformer and well-known LLMs. It will be incredibly helpful to have general knowledge of previously released large language models like LLaMA. I will also add my own interpretation of the DeekSeek model in this story.

Let's dive into the new features of model architecture step by step.

2. Model Architecture

First of all, we will investigate the core architecture of DeekSeek-V3 model. The DeekSeek-V3 model has inherited most parts of model from previous V2 model. These parts of model were elaborated more in V2 paper, but it is worth noting the principle how V3 model was built upon. While they used the structure of the ordinary transformer block like Llama, its attention and Feed-Forward Network were more sophisticated to boost the model performance. The overview of the Transformer block is as shown in the following diagram.

The two main components are Multi-Head Latent Attention(MLA) and DeepSeekMoE.

- 2.1 Multi-Head Latent Attention(MLA)

What is Multi-Head Latent Attention(MLA)? You might noticed that "Latent" is only additional word to conventional attention module. MLA improved the speed and memory usage in the attention block by compressing the input vector. From a data analysis perspective, the data can be compressed into a lower dimension while preserving the information it contains. One of the well-known techniques is Principal component analysis (PCA), which reduces the dimension of the data and maintains variance to retain its information. In latent diffusion model, the input data is compressed by variational autoencoder and reconstructed in initial dimension. The Multi-Head Latent Attention(MLA) applied this principle to compress and decompress the input data. By storing a compressed vector for the KV cache, the DeepSeek model can improve both speed by reducing data copying and memory efficiency by using a smaller compressed vector. The weight matrix for compression is additionally required, because human cannot compress it manually, but AI should learn it how the compression should be done. Applying RoPE to the compressed vector is not mathematically compatible, which was shown in detail in V2 paper, they used decoupled RoPE. As illustrated in the figure above, RoPE is applied to query and key, but also the query and key without RoPE. The RoPE-applied query and key are then concatenated with their respective non-RoPE counterparts. Finally, the query and key are obtained as normal transformer block, then the computation of dot-product attention will not be different from original one. But we could reach this point with a more economical KV cache thanks to the lower dimension of data.

- 2.2 DeekSeekMoE

Secondly, you can note that Feed-Forward Network is unusual as it was split into a lot of experts, rather than one large FFN. They called it as DeekSeekMoE. Like humans in a group, the AI also needs to specialized in certain domain to improve the performance. Thus, the mixture of experts come into play here. Each expert can specialize in certain domain, in this case, the group of tokens that they are familiar with, instead of coping with entire range of tokens alone. Dependent on the input sequence(tokens), the certain experts are selected to be activated and they contribute to make output. Shared experts are generalist and are activated for all kind of tokens. Then it might be interesting to know by what algorithm we can select the experts? We need to assign a vector to each expert which determines the range of tokens(domain) that experts can deal with well. And we give score to each expert to check how similar the domain of expert and input token are. If the score is high, then we should select the expert and let them activated to make output. Well, it sounds quite simple. Let's see the math behind it.

e_i is a centroid vector. It is learned during training and represents the type of input tokens the expert specialized in. Each expert's centroid vector encodes the knowledge domain it specializes in.

u_t is input vector to FFN. The dot product $u_t^T e_i$ quantifies the similarity between the input vector u_t and the centroid (or domain) of expert e_i, effectively measuring the alignment of the input data with the expert's specialized domain. So, the $s_i = \text{Sigmoid}(u_t^T e_i)$ represents the score for each i-th expert, determining whether the expert should be selected. By gating value g_i, which select K_r experts with high score by Topk algorithm. We add all outputs of selected experts and shared experts, then we arrive to the final output.

2.3 Multi-Token Prediction

In a standard transformer, the model generates a token each time, and this new token is fed back into decoder as input. Since this way restricts the efficiency and the speed of convergence during training, many researchers have made effort to come up with a method to generate multiple tokens each time. DeepSeek improved the conventional way of Multi-Token Prediction(MTP). Instead of previous parallel MTP, DeepSeek decided sequential

MTP. They construct independent MTP modules, where the previous output of the Transformer block is concatenated into the subsequent MTP module, as illustrated in the following figure.

As shown in the figure, the structure of MTP modules is akin to RNN model. But, unlike RNN, which preserve hidden states of nodes, the MTP modules send output of prediction to the subsequent module. Even though a single Transformer block cannot generate multiple tokens, the entire system of MTP modules collectively enables multi-token prediction. As it compares additional tokens per prediction, it provides more information for weight updates during training, leading to more efficient learning and faster convergence. The model can proactively learn and prepare for the additional tokens.

In actual training, DeepSeek opted to generate only one additional token, presumably due to the computational cost which is caused by using many MTP modules. It necessitate the compromise between the benefits of MTP and computational cost. During inference, the MTP modules are discarded, generating only one token per prediction.

3. Infrastructure

3.1 DualPipe

Since the U.S. did not export great GPUs like the NVIDIA H100 to China, DeepSeek researchers had to devise innovative methods to accelerate model training using the weaker H800 GPUs. Since they succeeded, NVIDIA's stock price briefly plunged, as people believed that high-performance GPUs would no longer be necessary for training LLM. Because the DeepSeek model was trained on 2048 H800 GPUs, communication between GPUs accounts for large portion of training time. Therefore, enhanching networking between GPUs has to play crucial role to reduce training time. When we use many GPUs simultaneously, the GPUs have to wait for a certain amount of time until new data is copied from other GPU. This waiting time, which causes training inefficiencies, is known as a "bubble," and we should minimize it as much as possible. DeepSeek invented a innovative method to reduce bubble.

During model training, data flows through the model in forward and backward processes. In forward process, data goes from the input layer to the output layer. On the other hand, during the backward process data moves from the output layer to the input later, updating weights based on the information to minimize the loss. Prior to DeepSeek, researchers found that the backward process can be split into two processes, which are backward for input and backward for weight, in order to remove more bubbles. The backward for input is computation of the gradient of the loss with respect to the input data, whereas the backward for weight calculates gradient of the loss with respect to the weight. The backward for input must be completed ahead of the backward for weight, because it is necessary to compute the backward for weight. Mathematically, the chain rule is applied to the calculation of backpropagation, where the backward for input is used for the calculation of backward for weight.

In such process, it is certain that an enormous number of communications between GPUs is required. In order to reduce the number of communication, the DeepSeek's DualPipe combines the forward process and the backward for input by initiating training data from two devices in the opposite directions as illustrated in following figure.

The batch 0 is the initial data, which starts processing on the device 0 and continues on the subsequent devices. In a conventional training plan, the device 7 remains idle, waiting for the batch 0 to be copied onto it. However, DualPipe makes the device 7 start training with other batch data in the opposite direction. This allows us to combine them as a chunk and continuously copy them together on other devices to reduce communication between GPUs. With weaker H800 GPUs, they couldn't improve the speed of the GPUs, but reduce the communication between GPUs to accelerate the training.

3.2 Mixed precision training

Mixed precision training is already prevalent technique of LLM to improve training and memory efficiency while maintaining the model accuracy. In mixed precision training, it is critical task to find out which parts of model are less significant for the model accuracy and reduce the precision that parts. In DeepSeek-V3 model, the researchers have found that they should reduce precision in the parts of model where heavy computations are executed, such as matrix multiplication. In contrast, they preserved high precision for matrix addition and storing data, which are relatively lightweight computation. The mixed precision training of DeepSeek is shown in the following figure.

While reducing the precision by the method above, overflow and underflow arise as a impediment. If the numerical values are quantized in lower precision like FP8 format, the values are clipped to the certain representable range. While computation in lower precision, the values can easily exceed the range during the computation. Scaling the values can mitigate the overflow and underflow by adjusting the values and lead to more proper representation in limited range. But, static scaling, which applies fixed scaling factor to all values, can still cause overflow and underflow for many values. To cope with this issue, DeepSeek implemented Fine-Grained Quantization. In this method, the values are grouped, and each group has its own scaling factor. This approach allows the each group of values to have a more suitable scaling factor, by which the overflow and underflow can be averted.

Another issue of quantization is that the small errors can be accumulated and become more serious problem later. In order to avoid that a lot of values with error are summed and their errors are accumulated, intermediate values are copied in high precision, if the number of values reaches the interval. It means that some values are grouped, and their values are stored in high precision. Then, the errors of values aren't accumulated on a large scale, because the small group of values don't contribute to large error.

These two techniques to prevent quantization error are visualized in following figure.

4. Reinforcement Learning

After supervised fine-tuning, DeepSeek additionally implemented reinforcement learning. A reward model has to be built and trained for reinforcement learning, which gives feedback to the model and determine the direction of learning. The rule-based reward model(RM) and model-based reward model(RM) were employed.

The rule-based RM is applied to the questions with specific rules, such as math problems and LeetCode problems. In these domains, the specific rules are used to verify the correctness of the answers and the questions about logical reasoning are involved. However, for many questions, the answer cannot be verified by a specific rule. In those cases where no rule is provided, the model-based RM determines, whether the answer matches the ground-truth answer. Another innovative idea of DeepSeek is including the chain-of-thought to the reward, whereas conventional models only included final reward based on the answer.

DeepSeek-V3 model, as V2 model did, adopted Group Relative POlicy Optimization (GRPO). This GRPO algorithm maximizes the following objective by updating the policy model π.

Maximize this objective by updating the weights of the model based on the reward.

Advantage is defined as the normalized reward.

In LLM case, the policy model π is model itself, and θ is weights of the model. q is question and o is output of the model. We can interpret the policy model(LLM) outputs a probability distribution over tokens, where the policy π(o|q) is a probability of output o given the question q. Therefore, the policy model is LLM itself. If the output o is right answer, we should reinforce the probability of that model makes this output o. So we need to maximize π(o|q) by multiplying advantage(normalized reward). If the output o is correct, the advantage (reward) will be a positive value and the policy will be reinforced. Otherwise, it will be negative and π(o|q) should be minimized. Plus, we have a fine-tuned model as the initial base model and do not want it to go too far from this base model, which might cause model to forget basic language understanding and important knowledge that the model learned during pre-training and fine-tuning. To implement this safety concerns, GRPO algorithm used KL divergence and epsilon parameter. The KL divergence measures the difference between current policy model and reference policy model(initial base model). So the KL divergence term should be minimized to maximized the GRPO objective. And we pick minimum between the original policy and the clipped policy in (1-ε, 1+ε), by which the model cannot deviate too much from 1. So, the current policy cannot differ a lot from the old policy, restricting the effect of reinforcement learning. This GRPO algorithm based on rule-based and model-based reward model enhances model performance and reasoning capability.

5. Conclusion

DeepSeek-V3 model offered great opportunity for efficient training with cheaper GPUs. It is unclear that its performance exceeds the OpenAI model, but DeepSeek is way more economical to train and open-source model. AI researchers can directly use DeekSeek models and they can also implement the innovative ideas and designs in their own model, because the new methods of DeepSeek and source code are opened. Seemingly, the DeepSeek researchers have potential to come up with more advanced idea to improve the model performance and efficient training process. In AI development, a lower training cost almost always implies better model accuracy later on, as the data and model can easily be scaled up at a lower cost. I hope that the performance of a good AI model does not have to be undermined by the censorship and suppression of the Chinese government.