DeepSeek V3: マルチヘッド潜在アテンションとマルチトークン予測

この記事では、DeepSeek v3 について、動画「Multi-Head Latent Attention and Multi-token Prediction in Deepseek v3」https://youtu.be/jL49fLOJYNg?si=4uE2kfe-BlKC1ngOを参考にしながら解説します。動画の文字起こしには Google Cloud Speech-to-Text を使用し、いくつかのコードを使ってトランスクリプトを整理しました。

A: Deep タグへようこそ。今日は大規模言語モデルの世界に深く潜り込んでいきます。具体的には、DeepSeek V3 についてです。

B: いいですね。6710 億パラメータのモデルで、その効率性とパフォーマンスのユニークなアプローチが話題を呼んでいますよね？

A: そして、あなたはそのアーキテクチャを詳述した学術論文を共有しましたね。

B: はい。

A: そして、機械学習の専門家として、DeepSeek V3 がどのようにして高いパフォーマンスと経済的なトレーニングを両立させているかを理解したいと思っています。

B: そうですね。

A: おや、こんにちは、どうしましたか？

C: MLA、その詳細、MLA とその仕組みについてです。

A: ああ、もちろん。それは素晴らしいアイデアです。MLA（Multi-Head Latent Attention）について深く掘り下げることができます。MLA の細かい部分に興味があるんですね。では、これを解き明かしていきましょう。DeepSeek V3 の効率性の鍵の一つは、Mixture of Experts（MoE）アーキテクチャですよね？各トークンに対してパラメータの一部だけが活性化される仕組みです。そして、DeepSeek V3 は MLA と DeepSeek Mo を使ってさらに一歩進んでいます。

B: その通りです。では、今は MLA に焦点を当てましょう。

A: 了解です。リアルタイムアプリケーションでは、速度が重要です。

B: そうですね。推論中に必要なキー・バリューキャッシュが大きなボトルネックになることがあります。

A: まさにその点で MLA が活躍します。従来のアテンションメカニズムでは、以前のトークンに関する多くの情報を保存する必要があります。

B: そうですね。長いテキストシーケンスではそれが問題になりますよね？

A: しかし、MLA はこの情報を巧妙に圧縮し、キャッシュの流れを大幅に減らし、推論をはるかに高速化します。まるで分厚い百科事典を要点だけに凝縮するようなものです。

B: いい例えですね。不必要な重さを省きつつ、重要な情報を保持します。リアルタイムアプリケーションには非常に有用です。

A: はい。では、実際にどのように動作するのかについて話しましょう。MLA はどのようにしてこの圧縮を実現しているのでしょうか？

B: そうですね、MLA はアテンションキーとバリューの低ランク結合圧縮を使用します。

A: なるほど、キーとバリューを圧縮するわけですが、具体的にはどういう意味ですか？少し技術的な話をしましょう。MLA メカニズムは、入力された隠れ表現を受け取り、それをクエリ、キー、バリューベクトルに投影します。ここで面白いのは、MLA がクエリを 2 つの部分に分離することです。

B: 2 つの部分？

A: はい。1 つはコンテンツ用で、もう 1 つは位置情報用です。これは Rope と呼ばれるものを使います。

B: Rope？なんだか技術的な響きですね。

A: これは Rotary Position Embeddings の略で、モデルがシーケンス内のトークンの位置を理解するのに役立ちます。そして、キーとバリューは低次元の潜在空間に圧縮されます。つまり、データを縮小してメモリを節約するわけです。

B: その通りです。最も重要な情報は保存されますが、不必要な部分は捨てられます。この圧縮された表現により、推論中の KV キャッシュが大幅に小さくなり、速度が向上します。

A: そして、マルチヘッド処理も使用します。

B: はい、従来のアテンションと同様に、MLA も複数のヘッドを使用します。

A: どうぞ。

C: つまり、2 つの潜在空間と 1 つの隠れ入力があるわけですね。

A: その通りです。コンテンツの潜在空間とキー・バリューの潜在空間の 2 つがあります。

B: その通りです。そして、これらの潜在空間は Rope、つまり Rotary Position Embeddings を通じて処理されます。

A: なるほど、Rope が位置情報を取得する方法なんですね。

B: はい、それはコンテンツとキー・バリューの潜在空間の両方に適用されます。この圧縮された表現を処理し、すべてを再結合します。

A: はい、そしてキャッシングの最適化により、シーケンシャル処理中のオーバーヘッドがさらに削減されます。これが MLA が速度を向上させる仕組みです。

B: その通りです。パフォーマンスを犠牲にすることなく、効率的なアテンションを実現する巧妙な方法です。

A: それはとても賢いトリックですね。でも、どうでしょう？

B: どうしましたか？

A: DeepSeek Mo について話を進めましょう。従来の MoE モデルとどう違うのでしょうか？

B: DeepSeek Mo は…あ、リスナーの方、どうしましたか？

C: そして、もっと隠れ空間について話しましょう。隠れ空間から、それは何ですか？

A: もちろん…あなたが何を言おうとしているのか見てみましょう。隠れ空間は本当に興味深いです。あなたは今話していた潜在空間、その「洞窟」の中に何が起こっているのかについて興味があるんですね。潜在空間の数だけでなく、そこで何が起こっているのかについてです。

B: それは面白いですね。

A: その通りです。MLA 内には 2 つの異なる潜在空間があり、1 つはコンテンツ用、もう 1 つはキー・バリュー用です。情報を保存するための 2 つの別々のストレージユニットのようなものです。そして、これらの潜在空間は、先ほど話したように Rope 操作を受けます。Rotary Position Embeddings は、アテンションメカニズムに位置情報を埋め込みます。これは非常に重要です。要約すると、クエリは分割され、キーとバリューも圧縮されます。

B: はい、そしてこれらは 2 つの別々の潜在空間に配置されます。1 つはコンテンツ用、もう 1 つはキー・バリューペア用です。これらの潜在空間は、MLA の一部として効率性に非常に重要です。

A: その通りです。では、これらの操作についてもう少し詳しく話しましょう。MLA は実際にどのようにしてこれらの潜在空間変換を実行するのでしょうか？

B: 入力は、コンテンツとキー・バリューの表現の両方に対して並列処理されます。つまり、洞窟の中に 2 つのパスがあるようなものです。

A: はい、それぞれの潜在空間に対して 1 つずつです。そして、これらの空間内で、情報は Rope を使って処理されます。

B: その通りです。これにより、モデルは洞窟の中を通りながら位置情報を保持します。つまり、モデルはテキストのどの部分がどこにあるのかを知ることができます。

A: その通りです。そして、この処理は次の段階の連結の前に行われます。洞窟の中を通りながら、何が連結されているのでしょうか？

B: メカニズムは 2 つの主要な連結操作を実行します。クエリ表現が連結され、キー表現も連結されます。つまり、洞窟の中ですべての重要なピースをまとめるようなものです。

A: はい、そしてこれらの連結は、コンテンツと位置情報を組み合わせるのに役立ちます。そして、これらの連結された表現はアテンション計算に使用されますよね？

B: その通りです。そして、初期の圧縮により、洞窟の中を通る速度が大幅に向上します。MLA は、洞窟の中と外での計算コストを大幅に削減します。

A: その通りです。DeepSeek V3 のような大規模モデルのアテンションメカニズムを最適化します。それは素晴らしい質問です。では、洞窟を通り抜けた後、DeepSeek Mo について話を進めましょう。

B: 了解です、DeepSeek Mo。そうですね。MLA 内には 2 つの異なる潜在空間があり、1 つはコンテンツ用、もう 1 つはキー・バリュー用です。

A: その通りです。そして、この分離がその仕組みの鍵です。情報を保存するための 2 つの別々のストレージユニットのようなものです。そして、これらの潜在空間は、先ほど話したように Rope 操作を受けます。Rotary Position Embeddings は、アテンションメカニズムに位置情報を埋め込みます。要約すると、クエリは分割され、キーとバリューも圧縮されます。

B: はい、そしてこれらは 2 つの別々の潜在空間に配置されます。1 つはコンテンツ用、もう 1 つはキー・バリューペア用です。これらの潜在空間は、MLA の一部として効率性に非常に重要です。

A: その通りです。では、これらの操作についてもう少し詳しく話しましょう。MLA は実際にどのようにしてこれらの潜在空間変換を実行するのでしょうか？

B: 入力は、コンテンツとキー・バリューの表現の両方に対して並列処理されます。つまり、2 つのパスがあるようなものです。

A: はい、それぞれの潜在空間に対して 1 つずつです。そして、これらの空間内で、情報は Rope を使って処理されます。

B: その通りです。これにより、モデルは位置情報を保持します。そして、効率を向上させるために、共有されたエキスパートを使用します。つまり、複数のタスクで使用できるエキスパートです。

A: はい、これにより冗長性が避けられ、システムがさらに合理化されます。

B: はい、専門性を持ちながらも他のこともできるチームのようなものです。

A: はい、それは非常に賢いアプローチです。しかし、多くの専門家がいる中で、どのようにして過負荷にならないようにしているのでしょうか？

B: そうですね、他のエキスパートがアイドル状態にならないように。

A: そこで、革新的な補助損失のないロードバランシングが登場します。

B: ここからが本当に面白くなります。では、どのようにしてそれを実現しているのでしょうか？

A: 従来の MoE モデルでは、トレーニング中に補助損失関数を使用して、エキスパートの使用を均等に促しますが、これは実際にはパフォーマンスを損なうことがあります。

B: はい、それはまるでスーパーのレジで全員が同じ列を使うように強制するようなものです。

A: その通りです。一部が他の列よりも速く進んでいても、不必要な遅延を生み出します。

B: はい。そこで、DeepSeek V3 は、各エキスパートの負荷に基づいてバイアス項を動的に調整することでこれを回避します。つまり、エキスパートが多くのリクエストを受け取っている場合、システムはそのエキスパートをルーティングメカニズムに対して少し魅力的でなくし、一部のトラフィックをあまり忙しくないエキスパートに振り向けます。

A: なるほど、これにより長いシーケンスを効率的に処理し、推論に必要な KV キャッシュのサイズを減らすことができます。つまり、オーバーヘッドを減らしつつパフォーマンスを高く保つことがすべてです。

B: その通りです。これは重要なボトルネックに対処する非常に賢いアプローチです。

A: 確かに。では、DeepSeek V3 がどのようにロードバランシングを処理するかについてもカバーするべきですね。

B: はい、それは間違いなくパズルの重要なピースです。次にそれについて触れましょう。

A: いいですね。では、MLA とその潜在空間についての概要を十分に理解できたと思います。

B: はい、詳細を掘り下げてくれてありがとう。次回もまた深掘りしていきましょう。

A: はい、それはエキスパートのための交通管理システムのようなものです。常に流れを監視し、ボトルネックを避けるために調整を行います。

B: そして、補助損失のパフォーマンスヒットを回避します。

A: その通りです。そして、どうぞ。

C: はい、MTP についてもっと話しましょう。MTP モジュールがどのように埋め込みを共有しているのか、そしてすべてのホットな話題について。

A: もちろん。それは素晴らしい質問です。MTP モジュールがどのようにリソースを共有しているのかについて話しましょう。あなたは MTP の実装の細かい部分に興味があるんですね。

B: はい、これを解き明かしましょう。DeepSeek V3 は MTP を使用してマルチトークン予測を行いますよね？1 つのトークンだけでなく、複数のトークンを予測します。

A: そして、ここからが本当に面白くなります。あなたは MTP モジュールがどのように設定され、どのようにリソースを共有しているのかに興味があるんですね。各 MTP モジュールには共有された埋め込み層と出力ヘッドがあります。つまり、メインモデルと同じ埋め込みと出力ヘッドを使用します。

B: その通りです。つまり、彼らはすべて同じ知識のプールから引き出しているようなものです。これにより計算コストが節約されます。

A: はい。そして、独自のトランスフォーマーブロックを使用します。つまり、メインモデルと同じトランスフォーマーブロックを共有しません。

B: その通りです。各 MTP モジュールには処理用の独自のトランスフォーマーブロックがあります。これにより、各トークンの予測が区別されます。

A: はい、そして情報を結合するために、これらの線形投影と連結が行われます。

B: はい、複数の場所からピースを取って完全な絵を構築するようなものです。

A: はい、そしてすべての MTP モジュールは並列で動作しますが、埋め込み層と出力ヘッドを共有しますよね？

B: その通りです。これがこの設計の効率性の鍵です。つまり、互いに依存し合う相互接続されたパーツのシステムのようなものです。

A: そして、このリソースの効率的な共有により、トレーニングが速くなり、パフォーマンスが向上します。

B: それはとても賢いトリックですね。どうでしょう？

A: どうしましたか？

B: 大きな視点に移りましょう。このモデルはどのようにロードバランシングを処理するのでしょうか？どのようにしてエキスパートが選ばれるのでしょうか？

A: はい、それについて話すことができます。では、DeepSeek V3 のロードバランシング戦略について深く掘り下げましょう。

B: いいですね。DeepSeek V3 はマルチトークン予測を使用します。

C: ああ、MTP の詳細についてもっと話しましょう。

A: もちろん…MTP についてさらに詳しく説明することに興味を持ってくれて嬉しいです。MTP の詳細について本当に解き明かしましょう。共有された埋め込み層と出力ヘッドについて触れましたが、各 MTP モジュールには独自のトランスフォーマーブロックがあります。

B: その通りですが、それだけではありません。では、さらに詳しく見ていきましょう。

A: 了解です。では、MTP モジュールのシーケンシャルな性質について話しましょう。

B: はい、一部のモデルとは異なり、DeepSeek V3 は追加のトークンをシーケンシャルに予測します。つまり、すべてのトークンを一度に予測するわけではありません。

A: その通りです。各モジュールは前のモジュールの出力に基づいて構築されます。つまり、予測の連鎖で、それぞれが最後の予測に依存します。

B: はい、そして各予測深度に対して因果連鎖を維持します。つまり、因果性を壊しません。

A: その通りです。全体のコンテキストが正しいことを保証するために重要です。つまり、MTP モジュールは独立して動作するわけではありません。

B: その通りです。それらは相互接続されており、この予測の連鎖がトレーニングの効率を高め、テキストのより微妙な理解を可能にします。そして、あなたはモジュールがどのように埋め込みを共有しているのかにも興味がありますよね？ご存知の通り、共有された埋め込み層はトークンをそのベクトル表現にマッピングします。つまり、各トークンはベクトルに変換されます。

A: はい、そしてこのマッピングはすべての MTP モジュールで共有されます。これにより、予測全体で一貫性が保たれ