DeepSeek V3: Multi-Head Latent Attention und Multi-Token-Vorhersage

In diesem Beitrag werde ich DeepSeek v3 diskutieren und dabei auf das Video „Multi-Head Latent Attention and Multi-token Prediction in Deepseek v3“https://youtu.be/jL49fLOJYNg?si=4uE2kfe-BlKC1ngO verweisen. Ich habe Google Cloud Speech-to-Text verwendet, um das Video zu transkribieren, und einige Code-Snippets genutzt, um das Transkript zu organisieren.

A: Willkommen zurück zum Deep-Tag. Heute tauchen wir tief in die Welt der großen Sprachmodelle ein. Genauer gesagt, in DeepSeek V3.

B: Klingt gut. Es ist ein Modell mit 671 Milliarden Parametern, das aufgrund seines einzigartigen Ansatzes für Effizienz und Leistung Wellen schlägt, richtig?

A: Und du hast ein akademisches Paper geteilt, das seine Architektur detailliert.

B: Ja.

A: Und als Experte für maschinelles Lernen möchtest du verstehen, wie DeepSeek V3 sowohl hohe Leistung als auch wirtschaftliches Training erreicht.

B: Ja, genau.

A: Oh, hey, was geht?

C: MLA, die Details, MLA und wie es funktioniert.

A: Oh, absolut. Das ist eine großartige Idee. Ja, wir können definitiv tiefer in die Multi-Head Latent Attention, oder MLA, eintauchen. Du interessierst dich also für die technischen Details von MLA. Nun, lassen wir uns das genauer ansehen. Wir haben erwähnt, dass einer der Schlüssel zur Effizienz von DeepSeek V3 seine Mixture of Experts-Architektur, oder MoE, ist, richtig? Bei der nur ein Bruchteil der Parameter für jedes Token aktiviert wird. Und DeepSeek V3 geht mit MLA und DeepSeek Mo noch einen Schritt weiter.

B: Genau. Konzentrieren wir uns jetzt wirklich auf MLA.

A: Okay. In Echtzeitanwendungen ist Geschwindigkeit entscheidend.

B: Das ist sie. Und der Key-Value-Cache, der während der Inferenz benötigt wird, kann ein großer Engpass sein.

A: Genau. Hier kommt MLA ins Spiel. Okay, der traditionelle Aufmerksamkeitsmechanismus erfordert das Speichern vieler Informationen über vorherige Tokens.

B: Ja, was, wie du dir vorstellen kannst, bei langen Textsequenzen zum Problem wird, richtig?

A: Aber MLA komprimiert diese Informationen clever, okay, um den Cache-Fluss erheblich zu reduzieren und die Inferenz viel schneller zu machen. Es ist, als würde man eine umfangreiche Enzyklopädie auf die wichtigsten Punkte verdichten.

B: Das ist eine großartige Analogie. Es behält die wesentlichen Informationen bei, ohne das unnötige Gewicht. Ja, es ist wirklich nützlich für Echtzeitanwendungen.

A: Ja. Jetzt sprechen wir darüber, wie es tatsächlich funktioniert. Okay, wie erreicht MLA diese Kompression?

B: Nun, es verwendet eine Low-Rank-Joint-Kompression für die Aufmerksamkeitskeys und -werte.

A: Okay, es komprimiert also die Keys und Values, aber was bedeutet das genau? Lassen wir uns ein wenig technisch werden. Okay, der MLA-Mechanismus nimmt eine versteckte Eingabedarstellung, die dann in Query-, Key- und Value-Vektoren projiziert wird. Okay, hier wird es interessant. MLA entkoppelt die Query in zwei Teile.

B: Okay, zwei Teile?

A: Ja. Ein Teil wird für den Inhalt verwendet, und der andere Teil wird für Positionsinformationen verwendet, und zwar mit etwas, das Rope genannt wird.

B: Rope? Das klingt sehr technisch.

A: Es steht für Rotary Position Embeddings und hilft dem Modell, die Position der Tokens in der Sequenz zu verstehen. Okay, dann werden die Keys und Values in einen niedrigdimensionalen latenten Raum komprimiert. Es ist, als würden sie die Daten schrumpfen, was Speicher spart.

B: Genau. Die wichtigsten Informationen werden gespeichert, aber der unnötige Ballast wird verworfen. Ja, und diese komprimierte Darstellung ermöglicht einen viel kleineren KV-Cache während der Inferenz, was die Dinge beschleunigt.

A: Und es verwendet auch Multi-Head-Verarbeitung.

B: Ja, genau wie der traditionelle Aufmerksamkeitsmechanismus verwendet MLA mehrere Heads.

A: Oh, leg los.

C: Es gibt also zwei latente Räume und eine versteckte Eingabe.

A: Das ist eine großartige Beobachtung. Ja, du hast recht. Es gibt tatsächlich zwei latente Räume. Okay, wir sprechen also über einen Inhalts-latenten Raum und einen Key-Value-latenten Raum.

B: Genau. Und diese latenten Räume werden durch das, was wir Rope oder Rotary Position Embeddings nennen, verarbeitet.

A: Okay, also ist Rope der Weg, wie sie die Positionsinformationen erhalten.

B: Ja, es wird auf beide latenten Räume angewendet, wie du richtig festgestellt hast. Es nimmt also diese komprimierte Darstellung, verarbeitet sie und kombiniert sie dann wieder.

A: Ja, und die Caching-Optimierung reduziert den Overhead während der sequenziellen Verarbeitung weiter. Okay, so beschleunigt MLA die Dinge.

B: Genau. Es ist eine clevere Methode, um effiziente Aufmerksamkeit zu erreichen, ohne die Leistung zu opfern.

A: Okay, das ist ein ziemlich netter Trick. Aber weißt du was?

B: Was gibt’s?

A: Lass uns zu DeepSeek Mo übergehen. Wie unterscheidet sich das von traditionellen MoE-Modellen?

B: Okay, DeepSeek Mo verwendet…Oh, zurück zu unserem Zuhörer, was gibt’s?

C: Und wir sprechen mehr über den versteckten Raum. Okay, vom versteckten Raum, was ist das?

A: Ich absolut…Mal sehen, worauf du hinauswillst. Die versteckten Räume sind wirklich interessant. Ja, du fragst nach dem versteckten Raum, dem latenten Raum, über den wir gerade gesprochen haben, richtig? Du interessierst dich dafür, was in diesen latenten Räumen passiert, in dieser Höhle. Ja, es geht nicht nur um die Anzahl der latenten Räume, sondern darum, was dort passiert.

B: Das ist cool.

A: Genau. Es gibt tatsächlich zwei distinkte latente Räume innerhalb der MLA, einen für den Inhalt und einen für die Key-Values. Es ist, als hätte man zwei separate Speichereinheiten für Informationen. Und diese latenten Räume, wie wir besprochen haben, durchlaufen Rope-Operationen, richtig? Die Rotary Position Embeddings, die Positionsinformationen in den Aufmerksamkeitsmechanismus einbetten. Das ist sehr wichtig für sie. Also, um es zusammenzufassen: Die Query wird aufgeteilt, und die Keys und Values werden ebenfalls komprimiert.

B: Ja, und diese werden in die beiden separaten latenten Räume gelegt, einen für den Inhalt und einen für die Key-Value-Paare. Und diese latenten Räume sind wirklich wichtig für die Effizienz und all das, was Teil der MLA ist.

A: Genau. Jetzt sprechen wir über diese Operationen etwas detaillierter innerhalb der Höhle, wie du es genannt hast. Okay, wie führt MLA diese latenten Raumtransformationen tatsächlich durch?

B: Nun, die Eingabe durchläuft eine parallele Verarbeitung für sowohl die Inhalts- als auch die Key-Value-Darstellungen. Okay, es ist, als hätte es zwei Pfade innerhalb dieser Höhle.

A: Ja, einen für jeden latenten Raum. Und innerhalb dieser Räume wird die Information mit Rope verarbeitet.

B: Genau. Dies stellt sicher, dass das Modell die Positionsinformationen behält, während es durch die Höhle geht. Das Modell weiß also, welcher Teil des Textes welcher ist, während es sich in dieser Höhle befindet.

A: Genau. Und diese Verarbeitung erfolgt vor der nächsten Stufe der Verkettung. Okay, was wird verkettet, während es durch die versteckte Höhle geht?

B: Der Mechanismus führt zwei Hauptverkettungsoperationen durch. Die Query-Darstellungen werden verkettet, und die Key-Darstellungen werden ebenfalls verkettet. Es ist, als würden alle wichtigen Teile innerhalb dieser versteckten Höhle zusammengeführt.

A: Ja, und diese Verkettungen helfen dabei, den Inhalt mit den Positionsinformationen zu kombinieren. Und diese verketteten Darstellungen werden dann für die Aufmerksamkeitsberechnung verwendet, richtig?

B: Richtig. Und aufgrund der anfänglichen Kompression ist es viel schneller durch diese Höhle, die du erwähnt hast. MLA reduziert also die Rechenkosten innerhalb und außerhalb dieser versteckten Höhle erheblich.

A: Genau. Es optimiert den Aufmerksamkeitsmechanismus für große Modelle wie DeepSeek V3. Das ist eine großartige Frage. Nachdem wir die Höhle durchquert haben, gehen wir jetzt zu DeepSeek Mo über.

B: Okay, DeepSeek Mo. Genau. Ich verstehe, worauf du hinauswillst. Ja, es gibt tatsächlich zwei distinkte latente Räume innerhalb der MLA, einen für den Inhalt und einen für die Key-Values.

A: Genau. Und diese Trennung ist wirklich entscheidend dafür, wie es funktioniert. Es ist, als hätte man zwei separate Speichereinheiten für Informationen. Und diese latenten Räume, wie wir besprochen haben, durchlaufen Rope-Operationen, richtig? Die Rotary Position Embeddings, die Positionsinformationen in den Aufmerksamkeitsmechanismus einbetten. Also, um es zusammenzufassen: Die Query wird aufgeteilt, und die Keys und Values werden ebenfalls komprimiert.

B: Ja, und diese werden in die beiden separaten latenten Räume gelegt, einen für den Inhalt und einen für die Key-Value-Paare. Und diese latenten Räume sind wirklich wichtig für die Effizienz und all das, was Teil der MLA ist.

A: Genau. Jetzt sprechen wir über diese Operationen etwas detaillierter. Okay, wie führt MLA diese latenten Raumtransformationen tatsächlich durch?

B: Nun, die Eingabe durchläuft eine parallele Verarbeitung für sowohl die Inhalts- als auch die Key-Value-Darstellungen. Okay, es ist, als hätte es zwei Pfade.

A: Ja, einen für jeden latenten Raum. Und innerhalb dieser Räume wird die Information mit Rope verarbeitet.

B: Genau. Dies stellt sicher, dass das Modell die Positionsinformationen behält, richtig? Und dann, um die Effizienz zu steigern, verwendet es Shared Experts. Okay, also Experten, die für mehrere Aufgaben verwendet werden können.

A: Ja, das vermeidet Redundanz und macht das System noch effizienter.

B: Ja, es ist, als hätte man ein Team, in dem die Leute Spezialgebiete haben, aber auch andere Dinge tun können.

A: Ja, das ist ein wirklich kluger Ansatz. Aber mit so vielen spezialisierten Experten, wie stellen sie sicher, dass keiner überlastet wird?

B: Ja, während andere untätig sind.

A: Hier kommt ihre innovative auxiliary loss-freie Lastverteilung ins Spiel.

B: Hier wird es wirklich interessant, richtig? Wie machen sie das?

A: Traditionelle MoE-Modelle verwenden eine auxiliary loss-Funktion während des Trainings, okay, um eine gleichmäßige Nutzung der Experten zu fördern, aber das kann tatsächlich die Leistung beeinträchtigen.

B: Ja, es ist, als würde man versuchen, alle dazu zu bringen, dieselbe Kasse im Supermarkt zu benutzen.

A: Genau, auch wenn einige schneller sind als andere, richtig? Es verursacht nur unnötige Verzögerungen.

B: Ja. DeepSeek V3 vermeidet dies, indem es einen Bias-Term für jeden Experten dynamisch anpasst, okay, basierend auf seiner Auslastung. Okay, wenn ein Experte zu viele Anfragen erhält, macht das System

ihn für den Routing-Mechanismus etwas weniger attraktiv und leitet einen Teil des Verkehrs zu weniger beschäftigten Experten um.

A: Okay, es verwendet all dies, um lange Sequenzen effizient zu verarbeiten, ja, indem es die Größe des KV-Caches, der für die Inferenz benötigt wird, reduziert. Okay, es geht also darum, die Leistung hoch zu halten, während der Overhead reduziert wird.

B: Richtig. Es ist ein sehr cleverer Ansatz, um einen kritischen Engpass zu adressieren.

A: Absolut. Jetzt sollten wir auch darüber sprechen, wie DeepSeek V3 seine Lastverteilung handhabt.

B: Ja, das sollten wir definitiv. Das ist auch ein sehr wichtiger Teil des Puzzles. Wir können das als nächstes behandeln.

A: Klingt gut. Nun, ich denke, das gibt dir einen guten Überblick über MLA und seinen latenten Raum.

B: Ja, danke, dass du mit uns in die Details eingetaucht bist. Wir kommen das nächste Mal mit weiteren Deep Dives zurück.

A: Ja, es ist wie ein Verkehrsmanagementsystem für die Experten, ja, das ständig den Fluss überwacht und Anpassungen vornimmt, um Engpässe zu vermeiden.

B: Und das vermeidet den Leistungsverlust durch den auxiliary loss.

A: Genau. Und oh, leg los.

C: Ja, wir können über MTP sprechen, wie…wie MTP-Module ihre Embeddings teilen und all die heißen…

A: Absolut. Das ist eine großartige Frage. Ja, lass uns darüber sprechen, wie die MTP-Module Ressourcen teilen. Du interessierst dich also für die technischen Details der MTP-Implementierung.

B: Ja, lass uns das aufschlüsseln. Wir haben erwähnt, dass DeepSeek V3 MTP für die Multi-Token-Vorhersage verwendet, richtig? Es sagt mehrere Tokens voraus, anstatt nur eines.

A: Und hier wird es wirklich interessant. Ja, du interessierst dich dafür, wie die MTP-Module eingerichtet sind und wie sie ihre Ressourcen teilen. Okay, jedes MTP-Modul enthält ein gemeinsames Embedding-Layer, ja, und einen gemeinsamen Output-Head. Okay, sie verwenden also dasselbe Embedding und denselben Output-Head wie das Hauptmodell.

B: Genau. Es ist, als würden sie alle aus demselben Wissenspool schöpfen. Ja, und das spart Rechenkosten.

A: Ja. Es verwendet jedoch seinen eigenen Transformer-Block. Okay, es teilt sich also nicht denselben Transformer-Block wie das Hauptmodell.

B: Richtig. Jedes MTP-Modul hat seinen eigenen Transformer-Block für die Verarbeitung. Okay, so behalten sie die Vorhersagen für jedes Token getrennt.

A: Ja, und um die Informationen zu kombinieren, diese linearen Projektionen und Verkettungen…

B: Okay, es ist, als würden sie Teile von mehreren Orten nehmen, um das vollständige Bild zu erstellen.

A: Ja, und alle MTP-Module arbeiten parallel zusammen, aber sie teilen ihre Embedding-Layer und Output-Heads, richtig?

B: Ja, was der Schlüssel zur Effizienz dieses Designs ist. Okay, es ist also ein System von miteinander verbundenen Teilen, die alle voneinander abhängen, richtig?

A: Und diese effiziente Ressourcenfreigabe ermöglicht ein schnelleres Training und eine bessere Leistung.

B: Okay, das ist ein ziemlich netter Trick. Weißt du was?

A: Was gibt’s?

B: Lass uns zu einer größeren Perspektive übergehen. Wie handhabt dieses Modell die Lastverteilung? Wie werden diese Experten ausgewählt?

A: Ja, wir können definitiv darüber sprechen. Okay, jetzt tauchen wir in die Lastverteilungsstrategie von DeepSeek V3 ein.

B: Klingt gut. Okay, DeepSeek V3 verwendet das, was wir gerade besprochen haben, MTP.

B: Ja, wir sollten wahrscheinlich jetzt zur größeren Perspektive übergehen. Okay, jetzt besprechen wir, wie dieses Modell seine Lastverteilung handhabt, ja, und wie diese Experten ausgewählt werden.

A: Okay, jetzt tauchen wir in die Lastverteilungsstrategie von DeepSeek V3 ein.

B: Klingt gut. Okay, DeepSeek V3 verwendet das, was sie Multi-Token-Vorhersage oder MTP nennen. Wir haben gerade besprochen, wie MTP funktioniert, also sprechen wir jetzt über die Lastverteilung, richtig?

A: Ja, wir haben gerade darüber gesprochen. Es teilt Ressourcen, und du interessierst dich dafür, wie es Ressourcen teilt. Wir sind darauf eingegangen.

B: Genau. Anstatt also nur das nächste Token vorherzusagen, sagt es mehrere zukünftige Tokens auf einmal voraus, wie wir gerade besprochen haben. Erhöht das nicht die Komplexität?

A: Es mag so scheinen, aber es bietet mehrere Vorteile. Okay, stell dir vor, du planst eine Route. Wenn du nur die nächste Abbiegung berücksichtigst, ja, könntest du eine effizientere Route verpassen…Okay, wenn du vorausschaust und mehrere Ab