

DeepSeek V3 : Attention Latente Multi-Têtes et Prédiction Multi-Tokens

Dans cet article, je vais discuter de DeepSeek v3, en référence à la vidéo “Multi-Head Latent Attention and Multi-token Prediction in Deepseek v3”<https://youtu.be/jL49fLOJYNg?si=4uE2kfe-BIKC1ngO>. J’ai utilisé Google Cloud Speech-to-Text pour transcrire la vidéo et un peu de code pour organiser la transcription.

A : Bienvenue sur Deep tag. Aujourd’hui, nous allons plonger dans le monde des grands modèles de langage. Plus précisément, DeepSeek V3.

B : Ça a l’air intéressant. C’est un modèle de 671 milliards de paramètres qui fait parler de lui pour son approche unique en matière d’efficacité et de performance, n’est-ce pas ?

A : Et tu as partagé un article académique détaillant son architecture.

B : Oui.

A : Et en tant qu’expert en apprentissage automatique, tu cherches à comprendre comment DeepSeek V3 parvient à concilier haute performance et entraînement économique.

B : Oui, c’est ça.

A : Oh, salut, quoi de neuf ?

C : MLA, les détails, MLA et comment ça fonctionne.

A : Oh, absolument. C’est une excellente idée. Oui, nous pouvons certainement approfondir l’attention latente multi-têtes, ou MLA. Donc, tu es curieux des rouages de MLA. Eh bien, déballons tout cela. Nous avons mentionné que l’une des clés de l’efficacité de DeepSeek V3 est son architecture de mixture d’experts, ou MoE, où seule une fraction des paramètres est activée pour chaque token. Et DeepSeek V3 va encore plus loin avec MLA et DeepSeek Mo.

B : C’est exact. Alors concentrons-nous sur MLA pour l’instant.

A : D’accord. Donc, dans les applications en temps réel, la vitesse est critique.

B : C’est vrai. Et le cache clé-valeur nécessaire pendant l’inférence peut être un goulot d’étranglement majeur.

A : Exactement. C’est là qu’intervient MLA. Donc, le mécanisme d’attention traditionnel nécessite de stocker beaucoup d’informations sur les tokens précédents.

B : Oui, ce qui, comme tu peux l’imaginer, devient un problème avec de longues séquences de texte, n’est-ce pas ?

A : Mais MLA compresse astucieusement ces informations, réduisant ainsi considérablement le flux de cache et rendant l’inférence beaucoup plus rapide. C’est comme si on prenait une encyclopédie volumineuse et qu’on la condensait en ne gardant que les points clés.

B : C'est une excellente analogie. Il conserve les informations essentielles sans le poids inutile. Oui, c'est vraiment utile pour les applications en temps réel.

A : Oui. Maintenant, parlons de son fonctionnement. Comment MLA parvient-il à cette compression ?

B : Eh bien, il utilise une compression conjointe de faible rang pour les clés et les valeurs d'attention.

A : D'accord, donc il compresse les clés et les valeurs, mais qu'est-ce que cela signifie exactement ? Alors, entrons un peu dans la technique. Le mécanisme MLA prend une représentation cachée en entrée, qui est ensuite projetée en vecteurs de requête, clé et valeur. Maintenant, voici où ça devient intéressant. MLA découple la requête en deux parties.

B : Deux parties ?

A : Oui. Une partie est utilisée pour le contenu, et l'autre pour les informations de position en utilisant quelque chose appelé Rope.

B : Rope ? Ça a l'air très technique.

A : Ça signifie "rotary position embeddings", et cela aide le modèle à comprendre la position des tokens dans la séquence. Ensuite, les clés et les valeurs sont compressées dans un espace latent de dimension inférieure. C'est comme si on réduisait les données, ce qui économise de la mémoire.

B : Exactement. Donc, les informations les plus importantes sont conservées, mais le volume inutile est éliminé. Oui, et cette représentation compressée permet un cache KV beaucoup plus petit pendant l'inférence, ce qui accélère les choses.

A : Et il utilise également un traitement multi-têtes.

B : Oui, tout comme l'attention traditionnelle, MLA emploie plusieurs têtes.

A : Oh, vas-y.

C : Donc, il y a deux espaces latents et une entrée cachée.

A : C'est une excellente observation. Oui, tu as raison. Il y a en fait deux espaces latents. Donc, nous parlons d'un espace latent pour le contenu et d'un autre pour les paires clé-valeur.

B : Exactement. Et ces espaces latents sont traités par ce que nous appelons Rope, ou "rotary position embeddings".

A : D'accord, donc Rope est la façon dont ils obtiennent les informations de position.

B : Oui, il est appliqué à la fois aux espaces latents de contenu et de clé-valeur, comme tu l'as souligné. Donc, il prend cette représentation compressée, la traite, puis combine le tout.

A : Oui, et l'optimisation de la mise en cache réduit encore plus la surcharge pendant le traitement séquentiel. C'est ainsi que MLA accélère les choses.

B : Exactement. C'est une manière astucieuse d'obtenir une attention efficace sans sacrifier les performances.

A : D'accord, c'est un tour assez ingénieux. Mais tu sais quoi ?

B : Quoi donc ?

A : Passons à DeepSeek Mo. En quoi est-ce différent des modèles MoE traditionnels ?

B : D'accord, DeepSeek Mo utilise...Oh, retour à notre auditeur, quoi de neuf ?

C : Et nous parlons plus d'espace caché. D'accord, à partir de l'espace caché, c'est quoi ?

A : Je comprends...Voyons ce que tu veux dire. Les espaces cachés sont vraiment intéressants. Oui, tu demandes à propos de l'espace caché, l'espace latent dont nous parlions, n'est-ce pas ? Tu es curieux de savoir ce qui se passe dans ces espaces latents, cette "caverne". Oui, ce n'est pas seulement une question de nombre d'espaces latents, mais de ce qui s'y passe.

B : C'est cool.

A : Exactement. Il y a en effet deux espaces latents distincts dans MLA, un pour le contenu et un pour les paires clé-valeur. C'est comme avoir deux unités de stockage distinctes pour les informations. Et ces espaces latents, comme nous l'avons discuté, subissent des opérations Rope, n'est-ce pas ? Les "rotary positional embeddings", qui intègrent les informations de position dans le mécanisme d'attention. C'est très important pour eux. Donc, pour résumer, la requête est divisée, et les clés et les valeurs sont également compressées.

B : Oui, et elles sont placées dans les deux espaces latents distincts, un pour le contenu et un pour les paires clé-valeur. Et ces espaces latents sont vraiment importants pour l'efficacité et tout ce qui fait partie de MLA.

A : Exactement. Maintenant, parlons de ces opérations un peu plus en détail dans la "caverne", comme tu l'as dit. D'accord, comment MLA effectue-t-il ces transformations d'espace latent ?

B : Eh bien, l'entrée subit un traitement parallèle pour les représentations de contenu et de clé-valeur. Donc, c'est comme s'il y avait deux chemins dans cette caverne.

A : Oui, un pour chaque espace latent. Et dans ces espaces, les informations sont traitées en utilisant Rope.

B : C'est vrai. Cela garantit que le modèle conserve les informations de position pendant qu'il traverse la caverne. Donc, le modèle sait quelle partie du texte est laquelle pendant qu'il est dans cette caverne.

A : Exactement. Et ce traitement est effectué avant l'étape suivante de concaténation. D'accord, qu'est-ce qui est concaténé pendant qu'il traverse la caverne de l'espace caché ?

B : Le mécanisme effectue deux opérations de concaténation majeures. Les représentations de requête sont concaténées, et les représentations de clé sont également concaténées. C'est comme rassembler toutes les pièces importantes dans cette caverne de l'espace caché.

A : Oui, et ces concaténations aident à combiner le contenu avec les informations de position. Et ces représentations concaténées sont ensuite utilisées pour le calcul de l'attention, n'est-ce pas ?

B : Correct. Et grâce à la compression initiale, tout est beaucoup plus rapide dans cette caverne que tu as mentionnée. Donc, MLA réduit considérablement les coûts de calcul à l'intérieur et à l'extérieur de cette caverne cachée.

A : Exactement. Il optimise le mécanisme d'attention pour les grands modèles comme DeepSeek V3. C'est une excellente question. Maintenant, après avoir traversé la caverne, passons à DeepSeek Mo.

B : D'accord, DeepSeek Mo. C'est ça. Je vois où tu veux en venir. Oui, il y a en effet deux espaces latents distincts dans MLA, un pour le contenu et un pour les paires clé-valeur.

A : Exactement. Et cette séparation est vraiment clé pour son fonctionnement. C'est comme avoir deux unités de stockage distinctes pour les informations. Et ces espaces latents, comme nous l'avons discuté, subissent des opérations Rope, n'est-ce pas ? Les "rotary positional embeddings", qui intègrent les informations de position dans le mécanisme d'attention. Donc, pour résumer, la requête est divisée, et les clés et les valeurs sont également compressées.

B : Oui, et elles sont placées dans les deux espaces latents distincts, un pour le contenu et un pour les paires clé-valeur. Et ces espaces latents sont vraiment importants pour l'efficacité et tout ce qui fait partie de MLA.

A : Exactement. Maintenant, parlons de ces opérations un peu plus en détail. D'accord, comment MLA effectue-t-il ces transformations d'espace latent ?

B : Eh bien, l'entrée subit un traitement parallèle pour les représentations de contenu et de clé-valeur. Donc, c'est comme s'il y avait deux chemins.

A : Oui, un pour chaque espace latent. Et dans ces espaces, les informations sont traitées en utilisant Rope.

B : C'est vrai. Cela garantit que le modèle conserve les informations de position, n'est-ce pas ? Et pour améliorer l'efficacité, il utilise des experts partagés. Donc, des experts qui peuvent être utilisés pour plusieurs tâches.

A : Oui, cela évite la redondance et rend le système encore plus rationalisé.

B : Oui, c'est comme avoir une équipe où les gens ont des spécialités mais peuvent aussi faire d'autres choses.

A : Oui, c'est une approche vraiment intelligente. Mais avec autant d'experts spécialisés, comment s'assurent-ils qu'aucun ne soit surchargé ?

B : Oui, pendant que d'autres restent inactifs.

A : C'est là qu'intervient leur équilibrage de charge innovant sans perte auxiliaire.

B : C'est là que les choses deviennent vraiment intéressantes, n'est-ce pas ? Alors, comment font-ils ?

A : Les modèles MoE traditionnels utilisent une fonction de perte auxiliaire pendant l'entraînement pour encourager une utilisation équilibrée des experts, mais cela peut en fait nuire aux performances.

B : Oui, c'est comme essayer de forcer tout le monde à utiliser la même caisse au supermarché.

A : Exactement, même si certaines avancent plus vite que d'autres, n'est-ce pas ? Cela crée simplement des retards inutiles.

B : Oui. Donc, DeepSeek V3 évite cela en ajustant dynamiquement un terme de biais pour chaque expert en fonction de sa charge. Donc, si un expert reçoit trop de demandes, le système le rend légèrement moins

attrayant pour le mécanisme de routage, redirigeant une partie du trafic vers des experts moins occupés.

A : D'accord, donc il utilise tout cela pour gérer efficacement les longues séquences, oui, en réduisant la taille du cache KV nécessaire pour l'inférence. Donc, il s'agit de maintenir des performances élevées tout en réduisant la surcharge.

B : Exactement. C'est une approche très astucieuse pour résoudre un goulot d'étranglement critique.

A : Absolument. Maintenant, nous devrions également aborder la façon dont DeepSeek V3 gère son équilibrage de charge.

B : Oui, nous devrions certainement en parler. C'est aussi une pièce très importante du puzzle. Nous pouvons en parler ensuite.

A : Ça me semble bien. Eh bien, je pense que cela te donne une bonne vue d'ensemble de MLA et de son espace latent.

B : Oui, merci d'avoir plongé dans tous les détails avec nous. Nous reviendrons la prochaine fois avec d'autres plongées profondes.

A : Oui, c'est comme un système de gestion du trafic pour les experts, oui, surveillant constamment le flux et faisant des ajustements pour éviter les goulots d'étranglement.

B : Et cela évite la perte de performance due à la perte auxiliaire.

A : C'est exact. Et oh, vas-y.

C : Oui, nous pouvons parler de MTP, comment...comment les modules MTP partagent leur embedding et tout le reste...

A : Absolument. C'est une excellente question. Oui, parlons de la façon dont les modules MTP partagent les ressources. Donc, tu es curieux des détails de l'implémentation de MTP.

B : Oui, déballons cela. Donc, nous avons mentionné que DeepSeek V3 utilise MTP pour la prédiction multi-tokens, n'est-ce pas ? Prédire plusieurs tokens au lieu d'un seul.

A : Et c'est là que ça devient vraiment intéressant. Oui, tu t'intéresses à la façon dont les modules MTP sont configurés et comment ils partagent leurs ressources. D'accord, donc chaque module MTP comprend une couche d'embedding partagée, oui, et une tête de sortie partagée. Donc, ils utilisent le même embedding et la même tête de sortie que le modèle principal.

B : Exactement. Donc, c'est comme s'ils puisaient tous dans le même bassin de connaissances. Oui, et cela économise sur les coûts de calcul.

A : Oui. Maintenant, il utilise son propre bloc de transformateur. Donc, il ne partage pas le même bloc de transformateur que le modèle principal.

B : Correct. Chaque module MTP a son propre bloc de transformateur pour le traitement. Donc, c'est ainsi qu'ils gardent les prédictions distinctes pour chaque token.

A : Oui, et pour combiner les informations, ces projections linéaires et concaténations...

B : Donc, c'est comme prendre des morceaux de plusieurs endroits pour construire l'image complète.

A : Oui, et tous les modules MTP travaillent en parallèle, mais ils partagent leurs couches d'embedding et leurs têtes de sortie, n'est-ce pas ?

B : Oui, ce qui est essentiel pour l'efficacité de cette conception. Donc, c'est comme un système de pièces interconnectées qui dépendent toutes les unes des autres, n'est-ce pas ?

A : Et ce partage efficace des ressources permet un entraînement plus rapide et de meilleures performances.

B : D'accord, c'est un tour assez ingénieux. Tu sais quoi ?

A : Quoi donc ?

B : Passons à une vue d'ensemble. Comment ce modèle gère-t-il l'équilibrage de charge ? Comment ces experts sont-ils choisis ?

A : Oui, nous pouvons certainement en parler. D'accord, maintenant plongeons dans la stratégie d'équilibrage de charge de DeepSeek V3.

B : Ça me semble bien. D'accord, donc DeepSeek V3 utilise ce qu'ils appellent la prédiction multi-tokens.

C : Oh oui, nous parlons plus des queues MTP.

A : Absolument...Je suis content que tu sois intéressé par une plongée plus profonde dans MTP. Oui, nous pouvons certainement élaborer sur la prédiction multi-tokens. Donc, nous en avons parlé, mais détaillons vraiment les détails de MTP, n'est-ce pas ? Nous parlions de la couche d'embedding partagée et de la tête de sortie, oui, et que chaque module MTP a son propre bloc de transformateur.

B : Exactement, mais il y a plus que ça. Alors, allons-y.

A : D'accord, parlons de la nature séquentielle des modules MTP.

B : Oui, contrairement à certains modèles, DeepSeek V3 prédit des tokens supplémentaires de manière séquentielle. Donc, il ne prédit pas tous les tokens en même temps.

A : Correct. Chaque module s'appuie sur la sortie du module précédent. Donc, c'est