ROTOR

Beaumont, Olivier, olivier.beaumont@inria.fr, INRIA, Bordeaux

Eyraud-Dubois, Lionel, lionel.eyraud-dubois@inria.fr, INRIA, Bordeaux

Shilova, Alena, alena.shilova@inria.fr, INRIA, Bordeaux

Duclos, Rémi, remi.duclos@inria.fr, INRIA, Bordeaux

Résumé : L'apparition et l'utilisation de réseaux de neurones de plus en plus profonds associés à des données volumineuses engendre des problèmes matériels au sein des grappes de calculs actuelles. En effet, s'il existe des heuristiques capables de contourner un blocage mémoire, elles impliquent une perte d'information et/ou un allongement non négligeable du temps de calcul. ROTOR - Rematerializing Optimally with pyTORch – propose une gestion automatique et optimale de la mémoire durant la phase d'apprentissage d'un réseau de neurones.
Mots clés : Optimisation, Réseaux de Neurones, PyTorch, mémoire, programmation dynamique, open source.

1. Introduction

La phase d'apprentissage d'un réseau de neurones est une opération très gourmande en mémoire. Il peut ainsi être impossible de réaliser l'apprentissage lorsque le réseau est profond et les données d'apprentissage ont une très bonne définition, parce que les mémoires des processeurs graphiques sont trop petites. Cette limitation mémoire peut donc limiter la précision de l'apprentissage.

2. Méthodologie

Lors de l'apprentissage, les activations calculées pendant la phase *forward* sont stockées pendant une certaine période de temps avant leur réutilisation dans la phase *backward*. ROTOR propose de libérer l'espace occupé par un sous-ensemble de ces activations et de les re-calculer au moment opportun. Le choix des tenseurs re-calculés est déterminé optimalement par un algorithme dynamique en fonction de la mémoire disponible, de façon à engendrer un allongement minimal du temps de calcul. Le programme dynamique est codé en langage C et interfacé en Python; son utilisation est complètement transparente pour les utilisateurs de PyTorch.

3. Originalité / perspective

Diverses approches ont été conçues pour faire face au problème de limitation mémoire dans PyTorch:
• Réduction de la taille des données, induisant une perte d'information.
• Re-calcul systématique de toutes les variables, augmentant significativement le temps d'apprentissage.
• Migration des variables sur d'autres mémoires disponibles avec temps supplémentaire de migration et utilisation du bus PCI.

L’optimalité de la gestion de la mémoire (rematerialization) proposée par ROTOR a permis d’augmenter significativement le volume de données traitée par les réseaux les plus couramment utilisés comme présenté dans les articles référencés ci-dessous.

Le code de ROTOR est actuellement disponible sur le gitlab d’INRIA et a été soumis à une intégration dans les prochaines versions de PyTorch. Une technique pour combiner la rematérialisation avec la migration vers la mémoire du CPU a été publiée récemment et est en cours d’intégration dans ROTOR.

Références

- Olivier Beaumont, Lionel Eyraud-Dubois, Julien Herrmann, Alexis Joly, Alena Shilova. **Optimal checkpointing for heterogeneous chains: how to train deep neural networks with limited memory.** [Research Report] RR-9302, Inria Bordeaux Sud-Ouest. 2019. https://hal.inria.fr/hal-02352969

- Olivier Beaumont, Lionel Eyraud-Dubois, Alena Shilova. **Efficient Combination of Rematerialization and Offloading for Training DNNs.** NeurIPS 2021 - Thirty-fifth Conference on Neural Information Processing Systems, Dec 2021, Virtual-only Conference https://hal.inria.fr/hal-03359793