Optimisation de l'exécution d'un réseau de neurones sur un accélérateur IA

Les algorithmes d'apprentissage profond permettent de nos jours de traiter de nombreux problèmes complexes dans des champs disciplinaires variés tels que la vision par ordinateur, la reconnaissance automatique de la parole ou des applications à la santé. Le développement de ces techniques est en grande partie lié à l'essor des moyens de calcul qui permettent l'utilisation de réseaux de neurones beaucoup plus importants en taille.

Les téléphones mobiles et l'arrivée de dispositifs portables intégrant des applications autour de la réalité augmentée ou de la réalité virtuelle nécessitent le développement de dispositifs embarqués capables de supporter des algorithmes d'apprentissage avec un nombre de neurones important. Pour Yann Lecun, un des challenges actuel est d'être capable de concevoir des architectures qui pourront supporter les algorithmes d'inférence pour un réseaux de neurones multi-couche avec peu d'énergie consommée et une vitesse élevée [1].

Depuis quelques années, plusieurs équipes de recherche internationales et d'industriels ont développé des solutions pour répondre à ce problème [2]. Les architectures obtenues sont généralement composées d'un nombre fixe de processeurs parallèles qui effectuent des opérations de calculs élémentaires (MAC pour Multiply and Accumulate, muliplication et somme). Ces processeurs peuvent être identiques comme pour Eyeriss [2] ou Movidius ou posséder des niveaux de consommation différents comme l'architecture Big.LITTLE de Arm. Ils sont généralement reliés à une mémoire partagée rapide et très limitée en taille.

Ces architectures sont conçues pour exécuter des algorithmes d'inférence pour des réseaux de neurones multicouches. Les différentes couches considérées sont généralement de type « feed-forward » ce qui inclus les convolutions (CONV) et les fully connected (FC). Pour un réseau de neurones fixé, le nombre de calcul a effectuer est constant et l'exécution est complètement prédictible. Par contre, la durée d'une exécution sur une architecture dédiée dépend fortement des stratégies d'accès aux données. En effet, une exécution de ce type passe son temps à charger des données dans les processeurs et à écrire les résultats obtenus en mémoire. Ainsi, les performances de ces architectures matérielles sont liées directement à la limitation et à l'utilisation de la mémoire partagée [2].

L'idée générale pour accélérer une exécution est alors de stocker dans les mémoires locales des processeurs, ou dans la mémoire partagée rapide les valeurs qui sont utilisées plusieurs fois dans des calculs différents de sorte à éviter des transferts trop nombreux et non nécessaires avec la mémoire centrale et ré-utiliser les données disponibles dans les processeurs ou la mémoire partagée.

Les architectures IA développées sont classées en familles en fonction du type de données qui sont stockées à proximité des processeurs. La famille des architectures « Output Stationary » a été conçue pour minimiser les lectures et écritures des sommes partielles. Ces valeurs intermédiaires sont simplement stockées dans la mémoire locale des processeurs. Ce choix architectural a été pris par de nombreux concepteurs d'architectures (voir par exemple [3,4,5]). Les processeurs sont généralement connectés en chaîne, en grille, ou en tore, et peuvent communiquer des données aux voisins (comme les poids) sans repasser par la mémoire partagée ou la mémoire centrale.

Le but de cette thèse est de développer des stratégies pour exécuter un réseau de neurones à plusieurs couches sur une accélérateur d'IA de type « Output Stationary ». Le réseau de neurones en entrée sera caractérisé par la taille et les opérations des différentes couches. On étudiera dans un

premier temps des opérateurs classiques de type « feed-forward » (CONV, FC). La poursuite sur des opérateurs de type récurrent (par ex. LSTM) sera envisagée en fonction de l'avancement des travaux. L'accélérateur IA considéré sera décrit par son nombre de processeurs, la structure des connections entre processeurs, la taille et les durées d'accès aux différentes mémoires. Le but est de développer une boite à outil algorithmique pour construire un placement et un ordonnancement de l'ensemble des opérations associées au réseau de neurones en entrée sur l'accélérateur IA fixée de sorte à optimiser la durée de calcul et respecter l'ensemble des contraintes liées aux mémoires.

A notre connaissance, ce sujet a été peu abordé sous cet angle. En effet, les concepteurs d'architectures IA proposent un bon placement et un ordonnancement pour une architecture fixée sans prendre en compte les différentes couches (les calculs sont effectués couche par couche) et sans faire la preuve de l'optimalité de leurs choix justifiés en général uniquement par des considérations expérimentales [2]. Les solutions proposés sont donc des optima locaux.

D'autre part, de nombreux travaux plus formels ont été consacrés à la recherche d'ordonnancements avec des contraintes mémoires. Sethi [6] à démontré en 1970 que le problème général de l'allocation de d'un nombre fixé de registres pour un graphe de tâches était NP-complet. Ce problème a été depuis largement étudié pour des graphes quelconques (voir par exemple [7]). Dans le cas de l'optimisation de l'exécution de la méthode multifrontale sur une architecture parallèle, Agullo et al. [8] ont développé des stratégies de placement et d'ordonnancement qui à la fois maximisent le parallélisme de l'application tout en respectant les contraintes de mémoire. Lemaitre et al . [8] ont également développé des stratégies basées entre-autre sur le dépliage des nids de boucles et la transformation des flottants en entiers pour accélérer la factorisation de Cholesky sur une machine SIMD. Une des questions de cette thèse est d'étudier si ces types d'approches peuvent être considérés dans le cas de réseau de neurones multicouches.

Profil de l'étudiant : le candidat retenue doit être titulaire d'un master d'Informatique ou de mathématiques appliquées avec des compétences en informatique théorique ou en recherche opérationnelle.

Références :

[1] Y. LeCun, Deep Learning Hardware: Past, Present, and Future, IEEE International Solid- State Circuits Conference, ISSCC 2019, San Francisco, CA, USA, February 17-21, 2019.

[2] V. Sze, Y. Chen, T. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks", Synthesis Lectures on Computer Architectures, Morgan and Claypool publishers, 2020.

[3] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Cheng, O.Temam, ShiDianNao: shifting vision processing closer to the sensor, Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, June 13-17, 2015.

[4] B. Moons , M. Verhelst, A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets, 2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits), Honolulu, HI, USA, 2016.

[5]S.Gupta, A.Agrawal, K.Gopalakrishnan, P.Narayanan , Deep Learning with limited numerical precision, International conference on machine learning, 2015.

[6] R.Sethi, STOC'73, Proceeding of the fifth annual ACM symposium on Theory of computing, April 1973.

[7] D.Sbîrla, Z.Budimlic, V.Sarkar, PACT '14: Proceedings of the 23rd international conference on Parallel architectures and compilation, August 2014

[8] E.Agullo, P.R. Amestoy, A.Buttari, A.Guermouche, J-Y.L'Excellent, F-H. Rouet, Robust Memory-Aware Mappings for Parallel Multifrontal Factorizations, SIAM Journal on Scientific Computing, 38(3), 2016.

[9]F.Lemaitre, B.Couturier, L.Lacassagne, Cholesky factorization on SIMD multi-core architectures, J. of Systems Architecture, vol 79, 2017.