Apports des alternatives à la rétropropagation dans l'apprentissage des réseaux de neurones binaires

Ben Crulis*, Barthelemy Serres*, Cyril de Runz*, Gilles Venturini*

*LIFAT, Université de Tours, Tours
https://lifat.univ-tours.fr/, nom.prenom@univ-tours.fr
**CETU ILIAD3, Université de Tours, Tours,
https://iliad3.univ-tours.fr

Résumé. Les modèles de réseaux de neurones profonds utilisent actuellement des paramètres encodés avec des nombres flottants utilisant beaucoup d'espace mémoire au moment de l'inférence. Ces modèles devenant de plus en plus volumineux, il devient difficile d'envisager entraîner directement ces modèles sur des appareils portables tels que les smartphones. Les réseaux de neurones binaires promettent de réduire la taille des réseaux de neurones artificiels tout en diminuant le temps d'inférence et l'énergie consommée, permettant le déploiement de modèles plus puissants sur les appareils portables. Cependant, les réseaux de neurones binaires sont encore actuellement difficiles à entraîner en utilisant la rétropropagation classique. Nous fournissons des tests comparatifs pour 3 algorithmes dont la rétropropagation permettant d'entraîner des réseaux de neurones binaires sur MNIST ainsi que CIFAR-10. Les résultats montrent que les réseaux de neurones binaires peuvent être entraînés en utilisant des alternatives à la rétropropagation, et donner lieu à de meilleures performances.

1 Introduction

Les réseaux de neurones artificiels sont connus pour leur bonnes performances sur de nombreux types de tâches, mais au prix de coûteuses phases d'apprentissage préalables. De nos jours, la recherche se concentre sur le passage à l'échelle pour de plus gros modèles et jeux de données puisque l'ajout de paramètres supplémentaires a tendance à améliorer les performances. Cette tendance rend les modèles plus encombrant et lent à l'utilisation. Leur utilisation est, de ce fait, plus difficile sur des appareils contraints en mémoire et en capacité de calcul tels que les smartphones. Pourtant, déployer les modèles directement sur les smartphones promet plusieurs avantages tel que réduire les communications réseaux et tout en réduisant la dépendance à d'éventuels serveurs de calcul distant. La perspective d'avoir des algorithmes d'apprentissage moins coûteux et des modèles moins volumineux ouvre la possibilité d'effectuer l'entraînement ou le ré-entraînement des modèles directement sur les appareils des utilisateurs, ce qui amène de nouvelles possibilités de personnalisation et d'adaptation à l'utilisateur.

Les réseaux de neurones binaires ont été proposés pour rendre les modèles de réseaux de neurones plus économes en mémoire tout en améliorant leur vitesse d'inférence et leur coût

énergétique grâce à l'utilisation d'opérations binaires de bas niveau. Ils sont donc le choix naturel pour le déploiement de modèles sur des appareils contraints (embarqués) tels que les smartphones. Les réseaux de neurones binaires ont été utilisés par exemple pour la détection de piétons (Ojeda et al., 2020), et la reconnaissance d'activités humaines (Daghero et al., 2021).

Malheureusement, les réseaux de neurones binaires sont plus difficiles à entraîner et souffrent de performances inférieures comparés à leur équivalent continus. La méthode moderne pour entraîner les réseaux de neurones binaires a été introduite par (Hubara et al., 2016). De part les différentes hypothèses et estimations utilisées pour rendre cela possible, l'entraînement reste malgré tout toujours plus difficile et donne lieu à des performances légèrement dégradées comparés aux modèles classiques utilisant une plus grande précision numérique.

Cependant, de nouveaux algorithmes d'apprentissage aux performances proches de la rétropropagation ont récemment été mis au point (Nøkland, 2016; Frenkel et al., 2021), permettant d'en envisager l'utilisation pour entraîner des réseaux de neurones binaires. À notre connaissance, ceci est le premier travail de recherche à proposer utiliser ces algorithmes alternatifs pour entraîner des modèles binaires.

Cet article propose une adaptation des algorithmes alternatifs à la rétropropagation pour l'entraînement des réseaux de neurones binaires et des tests sur les jeux de données MNIST et CIFAR-10 pour en évaluer la viabilité en terme de performance de classification. Le code des expériences est disponible sur `https://github.com/BenCrulis/binary_nn`.

2 Réseaux de neurones binaires

Dans cette section nous présentons les caractéristiques principales de réseaux de neurones binaires (Simons et Lee, 2019; Qin et al., 2020; Zhao et al., 2020).

Contrairement aux réseaux de neurones classiques avec des paramètres codés en valeurs réelles, chaque paramètre d'un réseau de neurones binaire est codé avec un seul bit. Un bit peut encoder pour une paire de valeurs, habituellement -1 et 1 pour les réseaux de neurones binaires puisque cela permet de remplacer les multiplications de nombres flottants par des opérations logiques XNOR plus efficientes. La conséquence immédiate est que les réseaux de neurones binaires sont beaucoup plus petits que leur équivalents continus, on peut s'attendre à une réduction en taille d'un facteur 32 par rapport à un modèle encodé avec des paramètres sur 32 bits.

Entraînement des réseaux de neurones binaires par descente de gradient Choisir les paramètres d'un modèle directement en binaire est un problème combinatoire NP-Complet. Pour cette raison, la méthode retenue consiste à entraîner un modèle avec des paramètres continu comme support pour un modèle binarisé. L'algorithme de rétropropagation est modifié de manière à calculer un gradient des paramètres qui ne soit pas nul partout au moment de traverser des étapes de binarisation. C'est ici qu'intervient le *Straight-Through-Estimator* (STE) (Hinton, 2012) qui permet de calculer une approximation non nulle du gradient pour la fonction $signe$.

Le modèle est ensuite entraîné de manière classique, avec les modifications proposées dans (Hubara et al., 2016). Dans la passe avant, le modèle est binarisé en utilisant la fonction $signe$ qui permet de passer des paramètres continus aux binaires puis utilisé pour calculer la sortie. Dans la passe arrière de la rétropropagation, le STE choisi est utilisé pour calculer

un gradient non nul au niveau de chaque étape de binarisation, au niveau des activations des neurones ainsi qu'a l'étape de binarisation des poids, permettant de calculer un gradient des poids du modèle continu. Au déploiement du modèle, le modèle continu sert à calculer les poids binaires finaux et abandonné, permettant de profiter de la réduction de la taille pour le stockage et l'utilisation.

La difficulté à entraîner les réseaux de neurones binaires vient des approximations qui sont faites pour permettre l'entraînement par calcul du gradient. Avec la méthode de rétropropagation habituelle, les gradients traversent donc plusieurs approximations de manière successive proportionnellement au nombre de couches. Nous conjecturons que cette accumulation d'approximations pourrait être à l'origine d'une baisse de performance pour les modèles plus profonds.

3 La rétropropagation et ses alternatives

Rétropropagation L'algorithme de rétropropagation (*backpropagation* en Anglais, abrégé en BP) est la méthode d'apprentissage la plus commune pour l'entraînement des modèles de réseaux de neurones profonds. L'algorithme BP permet de calculer la modification des poids permettant de réduire la valeur de la fonction de coût par descente de gradient. La règle de dérivation en chaîne permet de calculer de manière récursive le gradient des poids pour chaque couche dans ce constitue la passe arrière. Cette méthode est illustrée dans la Figure 1a.

Alignement de rétroaction directe Nøkland (2016) propose une amélioration de l'alignement de rétroaction (FA) introduite par Lillicrap et al. (2016) appelée Alignement de rétroaction directe (*Direct Feedback Alignment* en Anglais, abrégée en DFA). DFA consiste à propager le signal d'erreur au niveau de la sortie du modèle à chaque couche du modèle directement grâce à des matrices de poids aléatoires. Contrairement à FA où le signal d'erreur traverse jusqu'à $K-1$ couches, K étant le nombre de couches, dans DFA le signal d'erreur traverse 0 couche en passe arrière. Comme le signal d'erreur évite chaque couche en aval, cette méthode permet l'entraînement de modèles très profonds (avec plus de 100 couches) la où BP échoue à converger. Cet algorithme est illustré dans la Figure 1b.

Propagation directe de labels aléatoires Les méthodes précédentes nécessitent de calculer la fonction de coût afin d'ensuite obtenir un signal d'erreur pour entraîner les couches. Ceci empêche en pratique de mettre à jour les paramètres du modèle tant que la passe avant n'est pas complétée, ce fait est appelé *verrouillage de la mise à jour* (*update locking* en anglais). Cependant, l'algorithme, appelé Propagation directe de labels aléatoires (*Direct Random Target Propagation* en anglais, abrégé en DRTP), a été récemment proposé pour traiter ce problème (Frenkel et al., 2021) pour les modèles de classification. Pour chaque exemple d'apprentissage, la classe correcte sert à sélectionner pour chaque couche un vecteur aléatoire unique à cette classe tout au long de l'entraînement grâce à une projection par une matrice aléatoire de manière similaire à DFA. Ce vecteur est réinterprété comme signal d'erreur pour la couche considérée, le reste du processus est ensuite identique à BP et DFA. Par conception, DRTP permet de mettre à jour les paramètres d'une couche sans attendre que les couches suivantes aient même calculé leur propre résultats intermédiaires. DRTP est donc en principe beaucoup plus économe en mémoire puisque les variables intermédiaires peuvent être libérées après utilisation. Cet algorithme est illustré dans la Figure 1c.

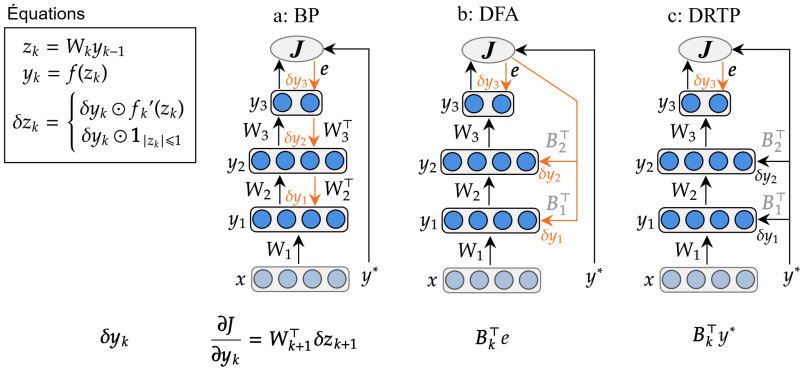


FIG. 1 – Schéma des algorithmes comparés dans les expériences. Les variables notées B_k sont des matrices aléatoires initialisée une seule fois avant le début de l'entraînement. Adapté de (Frenkel et al., 2021).

4 Algorithmes

Dans cette section nous présentons la méthode de binarisation utilisée.

La Figure 1 montre les différences entres les algorithmes. BP est le seul algorithme où le gradient traverse plusieurs couches en sens inverse, les deux autres envoient leur signal d'erreur en parallèle à chaque couche directement.

Afin de rendre les algorithmes d'entraînement compatibles avec la binarisation, nous modifions légèrement la façon dont les passes avant et arrière sont effectuées. D'abord, si la fonction d'activation est la fonction $signe$, un STE saturant est utilisé pour la passe arrière. Autrement dit, sa fonction dérivée est remplacée par le STE garantissant ainsi la propagation d'un gradient non nul en moyenne. Nous utiliserons dans la suite deux variants du STE. Soit δy_k le gradient à la couche k et δz_k le gradient estimé de la fonction $signe$. Une des variantes consiste à ignorer la dérivée de la fonction $signe$, dans ce cas $\delta z_k = \delta y_k$. Ce STE est non-saturant. Une autre variante introduite dans (Hubara et al., 2016) est équivalente à propager le gradient à travers une fonction $tanh$ dure ($Htanh(x) = clip(x, -1, 1)$), dans ce cas $\delta z_k = \delta y_k \odot \mathbf{1}_{|z_k|\leq 1}$. Ce STE prend la valeur 0 quand le neurone est saturé. Dans les expérimentations, nous utilisons le STE non-saturant pour la binarisation des poids et la version saturante pour la fonction $signe$ quand elle est utilisée en tant que fonction d'activation.

Dans la passe arrière, le signal d'erreur δy_k passe dans le STE de la fonction d'activation si elle est binaire, où bien dans la dérivée de la fonction d'activation $tanh$. Ceci nous donne le signal d'erreur δz_k qui est utilisé conjointement avec la variable sauvegardée y_{k-1} pour obtenir le gradient des paramètres δW_k après un passage dans le STE de la binarisation des poids. Dans le cas particulier de BP, le signal d'erreur δz_k est aussi propagé à la couche précédente d'index $k-1$ avec la formule suivante : $\delta y_{k-1} = (W_k^b)^\top \delta z_k$. Ce signal d'erreur traversera donc potentiellement plusieurs STE à la suite. Plus précisément, la couche k reçoit donc un signal ayant traversé $K-k$ STE, K étant le nombre de couches total. Les autres algorithmes quant à eux calculent l'erreur δy_k à partir de la fonction de coût à la dernière couche ou bien directement à partir des labels y^*. Dans ce cas un nombre constant de STE est traversé.

5 Expérimentations

Dans cette section nous présentons le protocole expérimental pour comparer les algorithmes sur trois jeux de données de classification d'images et présentons les résultats. Le but de ces expériences n'est pas d'obtenir un résultat supérieur à l'état de l'art mais de comparer les différences qualitatives et quantitatives des différents algorithmes quand nous passons d'un cadre continu à un cadre binaire.

Nous cherchons à répondre aux questions suivantes :

1. Quel est l'impact de la binarisation des activations pour chaque algorithme ?
2. Quel est l'impact de la binarisation des paramètres pour chaque algorithme ?
3. Quelle devrait être la méthode d'entraînement privilégiée pour un modèle entièrement binarisé, paramètres et activations ?

5.1 Protocole

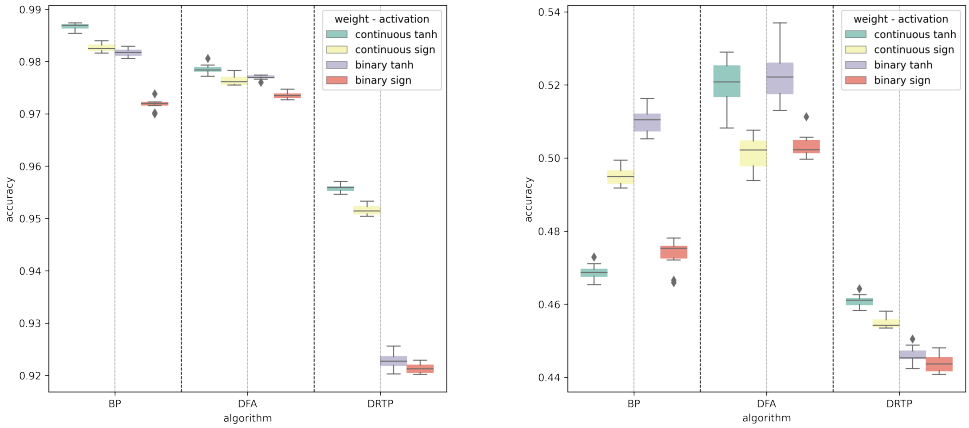
Nous proposons de mesurer les performance des modèles sur les données MNIST (Deng, 2012) et CIFAR-10 (Krizhevsky, 2009).

Toutes les couches et algorithmes sont entièrement ré-implémentés avec Numpy afin de s'assurer de fournir des conditions expérimentales identiques. Les termes de biais des couches denses sont retirés afin d'avoir la même architecture entre modèles continus et binaires. Le modèle est construit de manière à avoir 4 couches cachées ayant respectivement 700, 500, 300 et 200 neurones. La fonction de coût utilisée dans toutes les expériences est l'erreur quadratique moyenne. La taille de lot retenue est 128. Tous les algorithmes sont exécutés sur 100 epochs.

Nous entraînons les modèles avec chaque algorithme dans leurs versions continue et binaire. Nous séparons aussi les résultats par fonction d'activation ($tanh$ ou $signe$) afin d'évaluer l'impact de la binarisation des activations dans les couches cachées.

Nous effectuons d'abord une recherche exhaustive sur le taux d'apprentissage et l'échelle de l'initialisation des poids pour trouver la combinaison ayant le plus grand taux de classification correcte (*Accuracy*) en 100 epoch pour chaque triplet algorithme-binarisation-activation. Les valeurs considérées sont 10^{-4}, 10^{-5} et 10^{-6} pour le taux d'apprentissage et 10^{-3} ou 0.1 pour l'échelle d'initialisation des paramètres. Les paramètres sont initialisés avec une distribution uniforme avec l'échelle considérée.

L'architecture retenue a été choisie empiriquement de manière à avoir un nombre de couches suffisamment grand pour que les différences fondamentales entre algorithmes puissent affecter l'entraînement. Le modèle retenu a un total de 1110800 paramètres pour le modèle utilisé sur MNIST, sans compter les couches de normalisation par lot. Pour chaque tâche de classification, le modèle doit classifier les images parmi 10 classes équilibrées. Les tenseurs des images sont d'abord aplatis dans une représentation sous forme de vecteur. La dernière couche des modèles est toujours continue et possède toujours une fonction d'activation $tanh$. Cette dernière couche est toujours entraînée de la même manière pour tous les algorithmes à partir du gradient de la fonction de coût directement. Enfin, pour chaque algorithme nous répétons les expériences 10 fois avec le meilleur jeu d'hyperparamètres trouvé pour chaque combinaison de triplet algorithme-binarisation-activation.



(a) MNIST (b) CIFAR-10

FIG. 2 – Meilleur taux de classification sur les sous-ensembles de test.

5.2 Résultats

Dans cette section, nous présentons les résultats des expériences en terme de taux de classification correcte sur les différents jeux de données.

La Figure 2a montre les meilleurs taux de classification sur le jeu de test de MNIST. On observe que la transition des poids continus aux poids binaires ainsi que la transition des activations continues aux activations binaires tend à causer une augmentation du taux d'erreur dont l'ampleur dépend de l'algorithme considéré. DRTP est fortement impacté par le passage de poids continus aux poids binaires mais ne semble pas souffrir autant du passage des activations continues aux activations binaires. DFA est légèrement surpassé par BP sur le jeu de test, tout en étant supérieur dans le cas entièrement binarisé avec poids et activations binaires. Ce résultat est confirmé par un test ANOVA rapportant une valeur p de moins de 0.03% sur l'égalité des moyennes de ces deux groupes.

Les résultats sur CIFAR-10 sont rapportés dans la Figure 2b. Cette fois les résultats sont légèrement différents : BP est surclassé par DFA pour toutes les combinaisons de type de poids et de fonction d'activation mais reste plus performant que DRTP. Ici encore, dans le cas entièrement binaire, le modèle entraîné par DFA montre de bien meilleures performances que le modèle entraîné avec BP avec une valeur p de 7.9×10^{-13}.

6 Discussion

Quel est l'impact de la binarisation des activations pour les différents algorithmes ? Le passage de la fonction d'activation $tanh$ à la fonction signe semble responsable d'une légère baisse de performance pour tous les algorithmes et type de poids, binaires ou non. Cela s'explique certainement par l'utilisation du STE qui s'est malgré tout révélé très utile pour l'entraînement des réseaux de neurones binaires.

Quel est l'impact de la binarisation des poids pour les différents algorithmes ? La binarisation des poids semble pénaliser plus fortement les performances par rapport à la binarisation des activations. Cela est certainement dû à la saturation d'une partie des neurones puisque la binarisation des poids empêche effectivement les paramètres d'une couche de prendre de petites valeurs. Cette saturation implique que le gradient tende vers 0 ou même prenne la valeur 0 au passage de la dérivée de la fonction $tanh$ ou du STE respectivement. BP semble particulièrement affecté, plus probablement du fait que le gradient doive traverser plusieurs de ces neurones saturés de manière successive, empêchant le modèle d'apprendre des motifs utiles dans les premières couches.

Quelles méthode(s) devraient être retenue(s) pour entraîner des modèles ayant des poids et des activations binaires ? Ces expériences montrent que BP est très sensible à la binarisation aussi bien en poids qu'en activations. En revanche, DFA est bien moins perturbé y compris dans les versions entièrement binarisées du modèle.

Au final, parmi les trois algorithmes DFA semble être l'algorithme le plus prometteur pour entraîner des réseaux entièrement binarisés. Contrairement aux autres algorithmes, DFA est le seul algorithme à faire traverser à la fois un nombre constant de couches au signal d'apprentissage et à utiliser l'information apprise par les couches en aval indirectement par le calcul de l'erreur à la couche de sortie. Cette combinaison de qualités semble expliquer les performances de DFA dans le contexte de la binarisation.

7 Conclusion

Les réseaux de neurones binaires sont plus adaptés pour être utilisés sur les appareils embarqués par rapport à des modèles continus de par leur moindre coût mémoire et énergétique. L'entraînement actuel des réseaux de neurones binaires se base sur la rétropropagation et sur l'utilisation de STE. Néanmoins, plusieurs alternatives moins coûteuses que BP tels que DFA et DRTP ont été récemment proposés pour remplacer la rétropropagation mais n'avaient pas été exploitées pour entraîner les modèles binaires.

Nous avons proposé d'adapter DFA et DRTP pour les réseaux binaires et montré expérimentalement qu'il est possible d'égaler voire de surpasser les performances de la rétropropagation en utilisant DFA dans le cas de modèles ayant des poids et des activations binaires. Ces résultats permettent d'envisager que la rétropropagation pourrait ne pas être le meilleur choix quand il s'agit d'entraîner des réseaux de neurones binaires. Ces algorithmes alternatifs sont aussi par construction moins coûteux comparés à la rétropropagation et permettent donc d'imaginer de les utiliser directement pour entraîner des modèles sur appareils embarqués tels que les smartphones.

Références

Daghero, F., C. Xie, D. J. Pagliari, A. Burrello, M. Castellano, L. Gandolfi, A. Calimera, E. MacIi, et M. Poncino (2021). Ultra-compact binary neural networks for human activity recognition on risc-v processors. In *Proceedings of the 18th ACM International Conference on Computing Frontiers 2021, CF 2021*, pp. 3–11.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine 29*, 141–142.

Frenkel, C., M. Lefebvre, et D. Bol (2021). Learning without feedback : Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in Neuroscience 15*, 1–13.

Hinton, G. (2012). Neural networks for machine learning coursera video lectures.

Hubara, I., M. Courbariaux, D. Soudry, R. El-Yaniv, et Y. Bengio (2016). Binarized Neural Networks. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, et R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29 : NeurIPS*, Barcelona, Spain, pp. 4107–4115.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, Canada.

Lillicrap, T. P., D. Cownden, D. B. Tweed, et C. J. Akerman (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications 7*, 1–10.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 1045–1053.

Ojeda, F. C., A. Bisulco, D. Kepple, V. Isler, et D. D. Lee (2020). On-Device Event Filtering with Binary Neural Networks for Pedestrian Detection Using Neuromorphic Vision Sensors. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3084–3088.

Qin, H., R. Gong, X. Liu, X. Bai, J. Song, et N. Sebe (2020). Binary neural networks : A survey. *Pattern Recognition 105*, 107281.

Simons, T. et D. J. Lee (2019). A review of binarized neural networks. *Electronics 8*, 661.

Zhao, W., T. Ma, X. Gong, B. Zhang, et D. Doermann (2020). A review of recent advances of binary neural networks for edge computing. *IEEE Journal on Miniaturization for Air and Space Systems 2*, 25–35.

Summary

Current artificial neural networks are trained with parameters encoded as floating point numbers that occupy lots of memory space. Due to these models becoming more voluminous, it is becoming very difficult to consider training and using artificial neural networks on edge devices such as smartphones. Binary neural networks promise to reduce the size of deep neural network models as well as increasing inference speed while decreasing energy consumption and so allow the deployment of more powerful models on edge devices. However, binary neural networks are still difficult to train using the usual backpropagation algorithm. We provide experimental comparative results for three algorithms including the backpropagation baseline on the MNIST and CIFAR-10 datasets. The results demonstrate that binary neural networks can be trained using alternative algorithms to backpropagation and lead to better performance.