LITE: Light Inception avec des Techniques de Boosting pour la Classification de Séries Temporelles

Ali Ismail-Fawaz*, Maxime Devanne*, Stefano Berretti**, Jonathan Weber*
Germain Forestier*,***

*IRIMAS, Université de Haute-Alsace, Mulhouse France
{ali-el-hadi.ismail-fawaz,maxime.devanne,jonathan.weber,germain.forestier}@uha.fr,
**MICC, University of Florence, Florence, Italy
stefano.berretti@unifi.it,
***DSAI, Monash University, Melbourne, Australia

Résumé. Cet article est une traduction française de notre article intitulé "LITE : Light Inception with boosTing tEchniques for Time Series Classification" publié lors de la conférence internationale IEEE "Data Science and Advanced Analaytics (DSAA)" dans la session spéciale consacrée à l'apprentissage pour les données temporelles. Les modèles d'apprentissage profond se sont révélés être efficaces pour la classification des séries temporelles (CST). Les architectures de l'état de l'art, tout en produisant des résultats prometteurs sur l'archive UCR, présentent un nombre élevé de paramètres à entraîner. Cela peut conduire à un entraînement long avec une consommation d'énergie et une augmentation possible du nombre de *FLoating-points Operations Per Second* (FLOPS). Dans cet article, nous présentons une nouvelle architecture pour la CST : **Light Inception with boosTing tEchnique (LITE)** nécessitant seulement $2,34\%$ du nombre de paramètres du modèle InceptionTime, tout en préservant les performances. Cette architecture, qui ne compte que $9\ 814$ de paramètres à entraîner en raison de l'utilisation de convolutions séparables en profondeur (DWSC), est renforcée par trois techniques : le multiplexage, les filtres personnalisés et la convolution dilatée. L'architecture LITE, entraînée sur l'UCR, est $2,71$ fois plus rapide qu'InceptionTime et consomme $2,79$ fois moins de CO2 et d'énergie.

1 Introduction

Cet article est une traduction française de notre article intitulé "LITE : Light Inception with boosTing tEchniques for Time Series Classification" Ismail-Fawaz et al. (2023). La Classification des Séries Temporelles (CST) a été largement étudiée par les chercheurs ces dernières années. Certaines tâches de CST comprennent la classification de l'évaluation chirurgicale Ismail Fawaz et al. (2018), la détection de triche dans les jeux vidéo Pinto et al. (2021) *etc*.Grâce à la disponibilité de l'archive UCR Dau et al. (2019), la plus grande archive pour les ensembles de données CST, un nombre significatif de travaux a été réalisé dans ce domaine. Des modèles d'apprentissage profond ont été proposés dans le contexte des séries temporelles pour la classification Ismail Fawaz et al. (2019); Ismail-Fawaz et al. (2023), l'apprentissage de représenta-

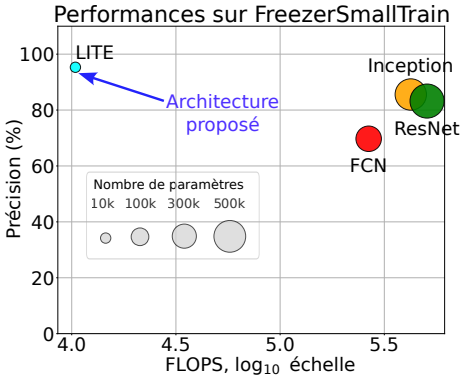

FIG. 1 – *Pour chaque modèle, la précision sur le jeu de données* FreezerSmallTrain *est présentée sur l'axe des y et le nombre de FLoating-point Operations Per Second (FLOPS) est présenté sur l'axe des x à l'échelle* $\log_{10}$. *Le diamètre des cercles représente le nombre de paramètres à entraîner du modèle.*

tion Ismail-Fawaz et al. (2023) *etc.*Même si les approches d'apprentissage profond ont prouvé leur efficacité pour la CST, elles présentent un grand nombre de paramètres à entraîner, ce qui conduit souvent à un temps d'entraînement long, un temps d'inférence long et une utilisation importante de stockage. Pour cette raison, certains travaux ont commencé à remettre en question le besoin d'une telle complexité dans les modèles d'apprentissage profond pour la CST tels que ROCKET et ses variantes Dempster et al. (2020); Tan et al. (2022).

Dans cet article, nous abordons la méthodologie de réduction de la complexité des modèles d'apprentissage profond, tout en préservant la performance de la tâche de CST. Nous faisons l'hypothèse qu'un modèle complexe et volumineux n'est peut-être pas nécessaire pour obtenir des bonnes performances sur l'archive UCR. Cependant, retirer des couches ou des paramètres pour réduire la complexité ne garantit pas la préservation des performances. Pour cette raison, l'architecture du réseau neuronal nécessite souvent des techniques supplémentaires afin de trouver un équilibre entre complexité et performance. Dans cet article, nous mettons en place des techniques qui ont déjà prouvé leur efficacité sur des modèles dédiés aux séries temporelles. Ces techniques sont les convolutions multiplexées, les convolutions dilatées et les filtres personnalisés Ismail-Fawaz et al. (2022). En combinant ces trois techniques avec une version modifiée d'un petit modèle moins complexe, le *Fully Convolutional Network* (FCN) Wang et al. (2017), nous proposons une nouvelle architecture nommée *Light Inception with boosTing tEchniques* (**LITE**).

Le modèle proposé utilise seulement $2,34\%$ du nombre de paramètres d'Inception, tout en étant compétitif avec les architectures de l'état de l'art. Par exemple, la Figure 1 montre que sur l'ensemble de données FreezerSmallTrain, la précision de classification de **LITE** est bien supérieure à d'autres approches avec beaucoup moins de paramètres à entraîner. Pour positionner l'architecture proposée parmi celles de l'état de l'art, nous comparons non seulement la précision, mais aussi le temps d'entraînement et le nombre de paramètres. Les principales contributions de ce travail sont :

— Une nouvelle architecture pour le CST, le modèle LITE avec seulement $2,34\%$ du nombre de paramètres d'Inception ;
— Des expériences approfondies montrant que LITE atteint des résultats compétitifs ;
— Une comparaison basée sur le nombre de paramètres à entraîner, le nombre de FLOPS, le temps d'entraînement, la consommation de CO2 et d'énergie.

2 Contexte et Travaux Connexes : Apprentissage Profond pour le CST

La Classification de Séries Temporelles (CST) a été largement étudiée dans la littérature. Dans cette section, nous présentons les travaux récents sur le CST utilisant des approches d'apprentissage profond. Le *Fully Convolutional Network (FCN)* a également été proposé dans Wang et al. (2017) qui utilise des opérations de convolution 1D. Dans ce modèle, l'algorithme de *backpropagation* trouve les meilleurs filtres pour extraire les caractéristiques des séries temporelles et classer correctement les échantillons. Dans ce modèle, les convolutions tiennent compte des dépendances temporelles dans les séries temporelles. Les auteurs de Wang et al. (2017) ont également proposé le modèle ResNet, qui utilise les connexions résiduelles pour résoudre le problème du *vanishing gradient*. Une étude comparative dans Ismail Fawaz et al. (2019) montre que l'utilisation de convolutions, en particulier ResNet, surpasse d'autres modèles qui utilisent la transformation multi-échelle ou des couches de pooling avec convolutions. Le modèle de référence en apprentissage profond pour le CST sur l'archive UCR Dau et al. (2019) est InceptionTime Ismail Fawaz et al. (2020), où les auteurs ont adapté le modèle Inception original pour les images aux séries temporelles. Il est important de noter que InceptionTime est un ensemble de cinq modèles Inception entraînés séparément. Plus récemment, la distillation de connaissances a également été abordée pour le modèle FCN Ay et al. (2022). Dans cette étude, les auteurs ont proposé une variante plus petite de FCN avec un nombre réduit de couches de convolution et de filtres à apprendre. De plus, les travaux de Ismail-Fawaz et al. (2022) ont proposé de créer manuellement certains filtres de convolution personnalisés au lieu de les générer aléatoirement comme dans Dempster et al. (2020); Tan et al. (2022). Avec ces filtres, les auteurs ont réduit le nombre de paramètres de FCN en proposant Hybrid FCN (H-FCN).

3 Approche proposée

3.1 DepthWise Separable Convolutions (DWSC)

Plusieurs Réseaux de Neurones à Convolution (*Convolutional Neural Networks* (CNNs)) ont été proposés pour la tâche de CST et ils ont tous prouvé leur supériorité par rapport à d'autres méthodes. Tout d'abord, nous présentons une manière d'appliquer des convolutions moins gourmandes en paramètres.

Le DWSC Howard et al. (2017) peut être divisé en deux phases : la *DepthWise Convoluton* (*Phase 1*) et la *PointWise Convolution* (*Phase 2*). Dans les convolutions standard, si l'échantillon d'entrée de longueur L a C_{in} canaux et que la sortie souhaitée est un espace avec C_{out} canaux utilisant un noyau de taille k, alors le nombre de paramètres appris est $C_{in} * C_{out} * k$.

Dans la phase 1, si la convolution est effectuée en utilisant un noyau de taille k, alors le nombre de filtres appris est C_{in}, et le nombre de canaux de sortie sera C_{in} (*comme l'entrée*). La phase 2 projette la sortie de la *DepthWise Convolution* dans un espace avec un nombre désiré de canaux C_{out}. Ceci est réalisé en appliquant une convolution standard avec C_{out} filtres de taille de noyau 1. Ainsi, en combinant ces deux phases, le nombre de paramètres appris dans un DWSC devient $C_{in} * k + C_{in} * C_{out}$.

Après avoir défini la technique pour utiliser les convolutions de manière plus optimisée en ce qui concerne le nombre de paramètres et de multiplications, d'autres techniques doivent également être définies. Ces techniques visent à minimiser l'impact de la réduction des paramètres dans les opérations de convolution expliquées ci-dessus.

3.2 Techniques de renforcement

Les techniques suivantes sont empruntées à des travaux précédents de la littérature.

— Multiplexage : La convolution multiplex a été proposée dans l'architecture d'Inception Ismail Fawaz et al. (2020). Son idée principale est d'apprendre en parallèle différentes couches de convolution ayant une taille de noyau différente.

— Dilatation des filtres : Les convolutions dilatées n'ont pas été très étudiées pour l'apprentissage supervisé profond sur CST, mais leur utilisation pour des approches *auto-supervisées* a démontré leur efficacité. La dilatation dans les filtres de convolution définit le nombre de cellules vides dans le noyau.

— Filtres Personnalisés : Utilisation des filtres crées manuellement comme pour H-FCN Ismail-Fawaz et al. (2022).

3.3 Light Inception with boosTing tEchniques (LITE)

Dans notre architecture proposée, nous réduisons le nombre de paramètres tout en préservant les performances. Ceci est obtenu en utilisant DWSC et les techniques de renforcement précédemment expliquées. Tout d'abord, des filtres personnalisés sont utilisés dans la première couche en parallèle à la première couche entraînable. Deuxièmement, dans cette première couche entraînable, une convolution de multiplexage est utilisée pour détecter différents motifs avec différentes caractéristiques (trois couches de convolution parallèles). Troisièmement, la deuxième et la troisième couche, des DWSC présentent l'utilisation de la dilatation dans leurs noyaux. Il est important de noter que pour la première couche, des convolutions standards sont utilisées à la place des DWSC. Ceci est dû au fait que la série temporelle d'entrée est univariée et que les DWSC n'apprendront qu'un seul filtre. Un résumé de l'architecture est donné à la Figure 2. On propose comme pour InceptionTime (ensemble de cinq Inception) Ismail Fawaz et al. (2020), LITETime un ensemble de cinq modèles de LITE.

4 Expérimentations

4.1 Jeux de données

L'archive UCR Dau et al. (2019) est le répertoire le plus important pour la CST et elle est disponible publiquement. L'archive contient 128 jeux de données pour des tâches de CST uni-

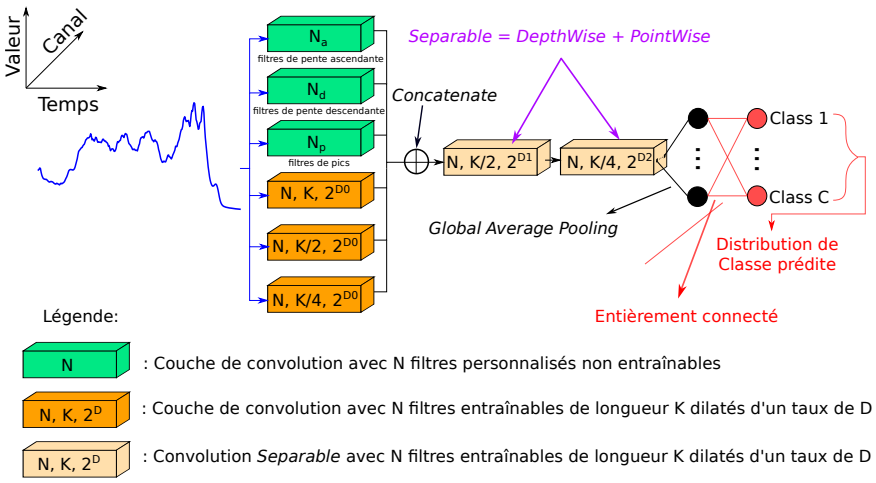

FIG. 2 – *L'architecture LITE proposée qui utilise des convolutions de multiplexage dans la première couche (trois blocs de convolution en orange) avec des filtres personnalisés (en vert). La deuxième et la troisième couche sont composées de DWSC (en beige). La dernière couche est suivie d'un regroupement moyen global (GAP) sur l'axe temporel et se termine par une couche entièrement connectée de classification pour estimer une distribution de classes.*

variées. Certaines tâches impliquent des données de séries temporelles d'électrocardiographie (ECG) et d'autres sont des observations de capteurs, *etc.*Chaque jeu de données est divisé en un ensemble d'entraînement et un ensemble de test. Les étiquettes sont disponibles pour tous les échantillons. Afin d'entraîner le modèle sur un ensemble de données normalisé, nous appliquons une z-normalisation sur tous les échantillons de manière indépendante. Cette technique de normalisation réduit les échantillons de séries temporelles en une séquence de moyenne nulle et de variance unitaire.

4.2 Détails de l'implémentation

Dans les expériences, nous avons pris en compte le temps d'entraînement, le temps d'inférence (temps de test), les émissions de CO2 et la consommation d'énergie [1]. Le modèle utilisé pour les tests est le meilleur modèle obtenu pendant l'entraînement, choisi en surveillant la perte d'entraînement. Le code source est disponible publiquement [2].

4.3 Résultats et Discussions

Tableau 1 présente la comparaison d'efficacité entre LITE et d'autres approches d'apprentissage profond. Tout d'abord, le tableau montre que le modèle le plus petit en termes de

1. https ://codecarbon.io/
2. https ://github.com/MSD-IRIMAS/LITE

TAB. 1 – *Comparaison entre les méthodes proposées avec FCN, ResNet et Inception.*

Modèles	Nombre de paramètres	FLOPS	Temps d'entraînement	Temps de test	CO2	énergie
Inception	420 192	424 414	145 267 secondes 1,68 jours	81 secondes 0,0009 jours	0,2928 g	0,6886 Wh
ResNet	504 000	507 818	165 089 secondes 1,91 jours	62 secondes 0,0007 jours	0,3101 g	0,7303 Wh
FCN	264 704	266 850	149 821 secondes 1,73 jours	27 secondes 0,00031 jours	0,2623 g	0,6176 Wh
LITE	**9 814**	**10 632**	**53 567 secondes 0,62 jours**	**44 secondes 0,0005 jours**	**0,1048 g**	**0,2468 Wh**

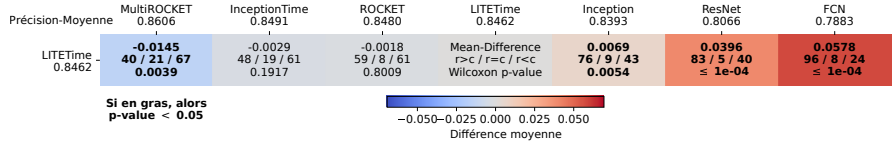

FIG. 3 – *La Matrice de Multi-Comparaison appliquée pour montrer les performances de LITETime par rapport à d'autres approches.*

nombre de paramètres est LITE avec $9,814$ paramètres. Cela est principalement dû à l'utilisation de DWSC au lieu de noyaux standards. Comparé à FCN, ResNet et Inception, LITE ne représente que $3,7\%$, $1,95\%$ et $2,34\%$ de leur nombre de paramètres respectivement. Deuxièmement, le modèle le plus rapide lors de la phase d'entraînement est LITE, avec un temps d'entraînement de $0,62$ jours. LITE est $2,79$ fois plus rapide que FCN, $3,08$ fois plus rapide que ResNet et $2,71$ fois plus rapide que Inception. Troisièmement, LITE est le modèle qui consomme la plus petite quantité de CO2 et d'énergie, soit $0,1048$ g et $0,2468$ kWh respectivement. Comparé aux autres approches, LITE présente le modèle le plus rapide et le plus écologique pour le CST par rapport à FCN, ResNet et Inception. Nous croyons, compte tenu des facteurs expliqués ci-dessus, que LITE peut être utilisé pour le déploiement de l'apprentissage profond pour le CST sur de petites machines telles que les téléphones mobiles. Dans ce qui suit, nous présentons les performances du modèle LITE proposé par rapport à l'état de l'art en termes de métrique de précision lors de l'évaluation sur les données de test.

Dans la suite, nous utilisons un nouvel outil d'évaluation, la *Matrice de Multi-Comparaison (MCM)*, qui est indépendante du nombre de méthodes à comparer Ismail-Fawaz et al. (2023). La MCM est présentée dans la Figure 3 pour comparer LITETime (ensemble de cinq LITE) à d'autres approches de la littérature. Chaque cellule dans le MCM présente le compte de Gain/Égalité/Perte ainsi que la moyenne de différence de performance entre les classifieurs et la P-valeur qui démontre si la différence de performance est significative. La MCM utilise la précision moyenne sur l'UCR comme métrique de classement. Comme présenté dans les MCM, LITETime fonctionnent mieux que FCN et ResNet en précision moyenne et est plus proche des performances d'InceptionTime, qui n'est pas significativement différent de LITETime (P-valeur élevée).

5 Conclusion

Dans cet article, nous avons abordé le problème de la classification des séries temporelles en réduisant le nombre de paramètres par rapport aux approches d'apprentissage profond existantes pour la classification des séries temporelles, tout en préservant les performances de InceptionTime. Nous avons présenté une nouvelle architecture pour la classification des séries temporelles, LITE, et avons évalué ses performances sur l'archive UCR. LITE n'a que $2,34\%$ du nombre de paramètres de Inception. Ce modèle est plus rapide que l'état de l'art en temps d'entraînement et d'inférence. Les résultats ont montré que l'utilisation de LITE nous permet d'obtenir des résultats proches de l'état de l'art sur l'archive UCR. Nous pensons que ce travail peut être le début de l'optimisation des architectures d'apprentissage profond dans le domaine des séries temporelles.

Remerciements

Ce travail a été soutenu par le projet ANR DELEGATION (subvention ANR-21-CE23-0014) de l'Agence Nationale de la Recherche française. Les auteurs souhaitent remercier le Centre de Calcul Haute Performance de l'Université de Strasbourg pour avoir soutenu ce travail en fournissant un support scientifique et un accès aux ressources informatiques. Une partie des ressources informatiques a été financée par le projet Equipex Equip@Meso (Programme Investissements d'Avenir) et le CPER Alsacalcul/Big Data. Les auteurs aimeraient également remercier les créateurs et fournisseurs de l'archive UCR.

Références

Ay, E., M. Devanne, J. Weber, et G. Forestier (2022). A study of knowledge distillation in fully convolutional network for time series classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Dau, H. A., A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, et E. Keogh (2019). The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica 6*(6), 1293–1305.

Dempster, A., F. Petitjean, et G. I. Webb (2020). Rocket : exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery 34*(5), 1454–1495.

Howard, A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, et H. Adam (2017). Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*.

Ismail-Fawaz, A., A. Dempster, C. W. Tan, M. Herrmann, L. Miller, D. F. Schmidt, S. Berretti, J. Weber, M. Devanne, G. Forestier, et al. (2023). An approach to multiple comparison benchmark evaluations that is stable under manipulation of the comparate set. *arXiv preprint arXiv :2305.11921*.

Ismail-Fawaz, A., M. Devanne, J. Weber, et G. Forestier (2022). Deep learning for time series classification using new hand-crafted convolution filters. In *2022 IEEE International Conference on Big Data (IEEE BigData 2022)*, pp. 1–8. IEEE.

Ismail-Fawaz, A., M. Devanne, J. Weber, et G. Forestier (2023). Enhancing time series classification with self-supervised learning. In *15th International Conference on Agents and Artificial Intelligence : ICAART 2023*. INSTICC : SciTePress.

Ismail-Fawaz, A., D. Maxime, B. Stefano, W. Jonathan, et F. Germain (2023). Lite : Light inception with boosting techniques for time series classification. In *International Conference on Data Science and Advanced Analytics (DSAA)*.

Ismail Fawaz, H., G. Forestier, J. Weber, L. Idoumghar, et P.-A. Muller (2018). Evaluating surgical skills from kinematic data using convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 214–221. Springer.

Ismail Fawaz, H., G. Forestier, J. Weber, L. Idoumghar, et P.-A. Muller (2019). Deep learning for time series classification : a review. *Data mining and knowledge discovery 33*(4), 917–963.

Ismail Fawaz, H., B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, et F. Petitjean (2020). Inceptiontime : Finding alexnet for time series classification. *Data Mining and Knowledge Discovery 34*(6), 1936–1962.

Pinto, J. P., A. Pimenta, et P. Novais (2021). Deep learning and multivariate time series for cheat detection in video games. *Machine Learning 110*(11-12), 3037–3057.

Tan, C. W., A. Dempster, C. Bergmeir, et G. I. Webb (2022). Multirocket : Multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 1–24.

Wang, Z., W. Yan, et T. Oates (2017). Time series classification from scratch with deep neural networks : A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE.

Summary

Deep learning models have been shown to be a powerful solution for Time Series Classification (TSC). State-of-the-art architectures, while conducting promising results on the UCR archive, present a high number of trainable parameters. This can lead to long training with a high CO2, energy consumption and possible increase in the number of FLoating-point Operation Per Second (FLOPS). In this paper, we present a new architecture for TSC, the **Light Inception with boosTing tEchnique (LITE)** with only 2.34% of the state-of-the-art model InceptionTime's number of parameters, while preserving performance. This architecture, with only $9,814$ trainable parameters due to the usage of DepthWise Separable Convolutions (DWSC), is boosted by three techniques: multiplexing, custom filters, and dilated convolution. The LITE architecture, trained on the UCR, is 2.78 times faster than InceptionTime and consumes 2.79 times less CO2 and energy.