The Journal of Open Source Software

ATHENA: A Fortran package for neural networks

Ned Thaddeus Taylor [1¶]

1 Department of Physics and Astronomy, University of Exeter, United Kingdom, EX4 4QL ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Rohit Goswami

Reviewers:

- @milancurcic
- @awvwgk
- @jrybarczyk

Submitted: 08 February 2024
Published: unpublished

Summary

In the landscape of modern Fortran programming, there exists a compelling need for neural network libraries tailored to the language. Given the extensive set of legacy codes built with Fortran, there is an ever-growing necessity to provide new libraries implementing on modern data science tools and methodologies. Fortran's inherent compatibility with high-performance computing resources, particularly CPUs, positions it as a language of choice for many machine learning problems.

The vast amount of computing capabilities available within current supercomputers worldwide would be an invaluable asset to the growing demand for machine learning and artificial intelligence. The ATHENA library is developed as a resource to bridge this gap; It provides a robust suite of tools designed for building, training, and testing fully-connected and convolutional feed-forward neural networks.

Statement of need

ATHENA (Adaptive Training for High Efficiency Neural Network Applications) is a Fortran-based library aimed at providing users with the ability to build, train, and test feed-forward neural networks. The library leverages Fortran's strong support of array arithmatics, and its compatibility with parallel and high-performance computing resources. Additionally, there exist many improvements made available since Fortran 95, specifically in Fortran 2018 (Reid, 2018) (and upcoming ones in Fortran 2023), as well as continued development by the Fortran Standards committee. All of this provides a clear incentive to develop further libraries and frameworks focused on providing machine learning capabilities to the Fortran community.

While existing Fortran-based libraries, such as neural-fortran (Curcic (2019)), address many aspects of neural networks, the focus on convolutional neural networks is drastically reduced. ATHENA is developed to handle both fully-connected and convolutional layers, including the ability to handle 3D data for convolutional layers (a domain sometimes underappreciated in comparison to its 3D counterpart). The ATHENA library is developed to handle diverse layer types, including fully-connected, Dropout, pooling, and convolution.

Notably, discussions with a spectrum of stakeholders have significantly influenced the development of ATHENA, placing paramount importance on accessibility and usability. This user-centric approach ensures that ATHENA is not just a library but a tool that seamlessly integrates with the evolving needs of the neural network community.

Features

A full list of features available within the ATHENA library, including available layer types, optimisers, activation functions, and initialisers, can be found on the repository's wiki.

ATHENA is developed to handle the following network layer types: batch normalisation (2D and 3D; Ioffe & Szegedy (2015)), convolution (2D and 3D), Dropout (Srivastava et al., 2014), DropBlock (2D and 3D; Ghiasi et al. (2018)), flatten, fully-connected (dense), pooling (2D and 3D; average and maximum).

The library can handle feed-forward networks with an arbirtray number of hidden layers and neurons (or filter sizes). There exist several activation functions, including Gaussian, linear, sigmoid, ReLU, leaky ReLU, tangent hyberbolic functions, and more. Optimiser functions include stochastic gradient decent (SGD), RMSprop, Adam, and AdaGrad. Network models can be saved to and loaded from files.

Ongoing research projects

The ATHENA library is being used in ongoing materials science research, with a focus on structural and materials property prediction.

Acknowledgements

The author thanks the Leverhulme for funding via Grant No. RPG-2021-086. The development of this code has benefitted through discussions with and contributions from many members of the Hepplestone research group, including Steven Paul Hepplestone, Francis Huw Davies, Harry McClean, Shane Davies, Ed Baker, Joe Pitfield, and Conor Price. Of particular note, Francis has provided contributions towards the development of code in some procedures focused on handling variables and files.

References

Curcic, M. (2019). A parallel fortran framework for neural networks and deep learning. *SIGPLAN Fortran Forum*, *38*(1), 4–21. https://doi.org/10.1145/3323057.3323059

Ghiasi, G., Lin, T.-Y., & Le, Q. V. (2018). DropBlock: A regularization method for convolutional networks. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 10750–10760. https://doi.org/10.5555/3327546.3327732

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 448–456. https://doi.org/10.5555/3045118.3045167

Reid, J. (2018). The new features of fortran 2018. *SIGPLAN Fortran Forum*, *37*(1), 5–43. https://doi.org/10.1145/3206214.3206215

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958. https://doi.org/10.5555/2627435.2670313