The Journal of Open Source Software

The nnlib2 library and nnlib2Rcpp R package for implementing neural networks

Vasilis N Nikolaidis[1]

1 University of Peloponnese

DOI: 10.21105/joss.02876

Software

- Review
- Repository
- Archive

Editor: Kakia Chatsiou

Reviewers:

- @schnorr
- @MohmedSoudy

Submitted: 22 October 2020
Published: 23 May 2021

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Artificial Neural Networks (ANN or NN) are computing models used in various data-driven applications. Such systems typically consist of a large number of processing elements (or nodes), usually organized in layers, which exchange data via weighted connections. An ever-increasing number of different neural network models have been proposed and used. Among the several factors differentiating each model are the network topology, the processing and training methods in nodes and connections, and the sequences utilized for transferring data to, within and from the model etc. The software presented here is a C++ library of classes and templates for implementing neural network components and models and an R package that allows users to instantiate and use such components from the R programming language.

Statement of need

A significant number of capable, flexible, high performance tools for NN are available today, including frameworks such as `Tensorflow` (Abadi et al., 2016) and `Torch` (Collobert et al., 2011), and related high level APIs including `Keras` (Chollet & others, 2015) and `PyTorch` (Paszke et al., 2019). Ready-to-use NN models are also provided by various machine learning platforms such as `H2O` (H2O.ai, 2020) or libraries, `SNNS` (Zell et al., 1994) and `FANN` (Nissen, 2003). Ready to use NNs are available in a large number of software packages (`WEKA`, `SPSS`, `Matlab`, `NeuroSolutions`, `Noesis` and many others). The R language (R Core Team, 2020), widely used for data analysis, includes package `nnet` (Venables & Ripley, 2002) in its standard packages, while a significant number of NN-related extension packages can be found in CRAN and other repositories. These include interfaces and bindings to aforementioned tools (`keras` (Chollet et al., 2017), `torch` (Falbel et al., 2020), `rTorch` (Reyes, 2020), `h2o` (LeDell & others, 2020), `RSNNS` (Bergmeir & Benítez, 2012)), as well as several other NN-specific packages that provide ready-to-use NN models (`deepnet` (Rong, 2014), `deepNN` (Taylor, 2020), RcppDL (Kou & Sugomori, 2014) etc.).

The `nnlib2` library adds another tool for NN experimentation and implementation. It is a collection of C++ base classes and class-templates for defining NN parts, components and entire models. It contains base classes (and related functionality) for each part of a NN (processing nodes, connections, layers, groups of connections etc.) and for NN models that contain such parts. The library does not focus (in its current state) on high-performance computing but on being a versatile basis on which new NN parts and models can be defined with consistent, readable, maintainable and reusable code. Although the software does contain some ready-to-use NN models and their related reusable NN parts (currently for versions of Back-propagation, autoencoder, Learning Vector Quantization and Matrix Associative Memory), its main purpose is to aid the creation of new custom NN parts and the experimentation with them in arbitrary NN models and configurations. Overall the library (and related R package):

1. Allows the implementation of NN parts and models from reusable components that follow a common interface thus can easily be modified, combined with each other etc, making them suitable for educational or experimentation purposes.
2. Provides a basis for implementation of different NN parts and models, including classic NNs.
3. Is versatile enough for experimentation with new, custom components, configurations and models.
4. Has no dependencies with other software; to build or employ the `nnlib2` parts and models only a standard C++ compiler is required, and the produced models are standalone, lightweight, highly portable and can be embedded in any C++ application.
5. Allows for the exact same NN parts (processing nodes, connections, layers, groups of connections etc.) defined using `nnlib2` to be employed, combined, manipulated and monitored in R, via package `nnlib2Rcpp` which includes the entire `nnlib2` library and also provides supporting R functions and modules for this purpose. Thus, new NN components can be developed entirely using R-related tools (such as Rtools and RStudio), and then be used in R or transferred to a pure C++ application if needed.

Discussion

As implied earlier, the `nnlib2` library may appeal to NN students and experimenters who seek a basis for implementing various NN models in C++ and take advantage of the versatility and direct control of the model that this approach allows. Furthermore, the library lets new NN definitions be written in code that is comparatively simple and self-explanatory; most of the provided base classes and templates correspond to typical NN components (processing nodes, layers of nodes, connections, sets of connections between layers, entire neural networks etc.) and closely follow the conceptual model often found in related bibliography such as (McCord-Nelson & Illingworth, 1991; Pao, 1989; Simpson, 1991; Theodoridis & Koutroumbas, 2008) and elsewhere. Typical component functionality and features are provided in these base definitions, while diverse, model-specific components are to be implemented in custom sub-classes. By following a common interface, components (or even entire NN models) can be reused in other models. Finally, the library presents a high degree of target and compiler independence, with early versions used in various solutions targeting different operating systems, mainly for research purposes, for example in (Vasilios Nikolaidis, 1999; V. N. Nikolaidis et al., 2013; Philippidis et al., 1999).

To take advantage of the R ecosystem, improve its usability and widen its audience, the `nnlib2` library has also been integrated into an R package (a process supported by `Rcpp` (Eddelbuettel et al., 2020)). The package, named `nnlib2Rcpp` (available on CRAN (Vasilis Nikolaidis, 2020b) and GitHub (Vasilis Nikolaidis, 2020a)) includes the entire `nnlib2` code as well as an additional R class (class NN) with methods that allow users to instantiate the `nnlib2` NN components they define, combine them in custom NN configurations and generally control, monitor and employ them from R. Finally, the package provides a small collection of predefined NN components and complete ready-to-use NN models (for versions of Back Propagation, autoencoder, supervised/unsupervised Learning Vector Quantization and Matrix Associative Memory neural networks).

References

Abadi, M., P.Barham, J.Chen, Z.Chen, A.Davis, J.Dean, M.Devin, S.Ghemawat, G.Irving, M.Isard, & others. (2016). Tensorflow: A system for large-scale machine learning. *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Bergmeir, C., & Benítez, J. M. (2012). Neural networks in R using the stuttgart neural network simulator: RSNNS. *Journal of Statistical Software*, *46*(7), 1–26. http://www.jstatsoft.org/v46/i07/

Chollet, F., Allaire, J., & others. (2017). *R interface to keras*. https://github.com/rstudio/keras; GitHub.

Chollet, F., & others. (2015). *Keras*. https://keras.io.

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. *BigLearn, NIPS Workshop*.

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Russell, N., Bates, D., & Chambers, J. (2020). *Rcpp: Seamless r and c++ integration*. https://CRAN.R-project.org/package=Rcpp

Falbel, D., Luraschi, J., & others. (2020). Package 'torch.' In *CRAN repository*. https://CRAN.R-project.org/package=torch

H2O.ai. (2020). *H2O: Scalable machine learning platform*. https://github.com/h2oai/h2o-3

Kou, Q., & Sugomori, Y. (2014). *RcppDL: Deep learning methods via rcpp*. https://CRAN.R-project.org/package=RcppDL

LeDell, E., & others. (2020). *h2o: R interface for the 'H2O' scalable machine learning platform*. https://CRAN.R-project.org/package=h2o

McCord-Nelson, M., & Illingworth, W. T. (1991). *A practical guide to neural nets*. Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201523760

Nikolaidis, Vasilios. (1999). *Non-destructive inspection (NDI) techniques for composite materials using unconventional pattern regognition methods (in greek)* [PhD thesis, University of Patras]. https://www.didaktorika.gr/eadd/handle/10442/11158?locale=en

Nikolaidis, Vasilis. (2020a). 'nnlib2Rcpp.' In *GitHub repository*. https://github.com/VNNikolaidis/nnlib2Rcpp

Nikolaidis, Vasilis. (2020b). Package 'nnlib2Rcpp.' In *CRAN repository*. https://CRAN.R-project.org/package=nnlib2Rcpp

Nikolaidis, V. N., Makris, I. A., & Stavroyiannis, S. (2013). ANS-based preprocessing of company performance indicators. *Global Business and Economics Review*, *15*(1), 49–58. https://doi.org/10.1504/gber.2013.050667

Nissen, S. (2003). *Implementation of a fast artificial neural network library (FANN)*. https://github.com/libfann/fann

Pao, Y. (1989). *Adaptive pattern recognition and neural networks*. Reading, MA (US); Addison-Wesley Publishing Co., Inc.

Paszke, A., Gross, S., & others. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Philippidis, T., Nikolaidis, V., & Kolaxis, J. (1999). Unsupervised pattern recognition techniques for the prediction of composite failure. *Journal of Acoustic Emission*, *17*(1-2), 69–81.

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Reyes, A. (2020). Package 'rTorch.' In *CRAN repository*. https://CRAN.R-project.org/package=rTorch

Rong, X. (2014). *Deepnet: Deep learning toolkit in r*. https://CRAN.R-project.org/package=deepnet

Simpson, P. K. (1991). *Artificial neural systems: Foundations, paradigms, applications, and implementations*. McGraw-Hill, Inc. ISBN: 0071053557

Taylor, B. (2020). *deepNN: Deep learning*. https://CRAN.R-project.org/package=deepNN

Theodoridis, S., & Koutroumbas, K. (2008). *Pattern recognition, fourth edition* (4th ed.). Academic Press, Inc. ISBN: 1597492728

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with s* (Fourth). Springer. http://www.stats.ox.ac.uk/pub/MASS4

Zell, A., Mache, N., Hübner, R., Mamier, G., Vogt, M., Schmalzl, M., & Herrmann, K.-U. (1994). SNNS (stuttgart neural network simulator). In J. Skrzypek (Ed.), *Neural network simulation environments* (pp. 165–186). Springer US. https://doi.org/10.1007/978-1-4615-2736-7_9