WekaDeeplearning4j: a Deep Learning Package for Weka based on DeepLearning4j

Steven Lang[a], Felipe Bravo-Marquez[b], Christopher Beckham[c], Mark Hall[d], Eibe Frank[e]

[a] *Technische Universität Darmstadt, Darmstadt Germany*
[b] *Department of Computer Science, University of Chile & IMFD, Santiago, Chile*
[c] *École Polytechnique de Montréal, Montréal, Canada*
[d] *Hitachi Vantara*
[e] *Department of Computer Science, University of Waikato, Hamilton, New Zealand*

Abstract

Deep learning is a branch of machine learning that generates multi-layered representations of data, commonly using artificial neural networks, and has improved the state-of-the-art in various machine learning tasks (e.g., image classification, object detection, speech recognition, and document classification). However, most popular deep learning frameworks such as TensorFlow and PyTorch require users to write code to apply deep learning. We present WekaDeeplearning4j, a Weka package that makes deep learning accessible through a graphical user interface (GUI). The package uses Deeplearning4j as its backend, provides GPU support, and enables GUI-based training of deep neural networks such as convolutional and recurrent neural networks. It also provides pre-processing functionality for image and text data.

Keywords: Deep Learning, Weka

1. Introduction

We present WekaDeeplearning4j[1], a tool for training and testing deep learning models implemented in Deeplearning4j[2] from within Weka [1], a widely used open-source machine learning workbench implemented in Java.

[1] `https://deeplearning.cms.waikato.ac.nz/`
[2] `https://deeplearning4j.org/`

The main goal of our tool is to make deep learning accessible without requiring users to write code—via Weka's GUI. At the most basic level, the GUI enables users to perform experiments using the following simple steps: 1) loading data in the Attribute-Relation File Format (ARFF)[3], 2) configuring a neural network architecture, 3) choosing an experimental protocol, and 4) running the experiment.

The WekaDeeplearning4j package supports fully connected feedforward networks, convolutional networks, and recurrent networks. Data loaders for standard tabular data, as well as image, text, and sequence data, are provided. It is also possible to train a neural network and use it as a feature extractor to provide suitable input data for another learning algorithm implemented in Weka, such as a support vector machine. Because the neural network predictors in the package are standard Weka "classifier" objects, they can be used and deployed in the same way as other types of predictive models generated by learning algorithms in Weka.

2. Problems and Background

Recent years have seen many breakthroughs achieved with deep neural networks, delivering state-of-the-art results in machine learning tasks such as image and document classification. Companies make use of the latest deep learning breakthroughs by benefiting from a quickly evolving software environment comprising different deep learning frameworks. Major frameworks are TensorFlow[4], PyTorch [2], Deeplearning4j, CNTK [3] and Caffe [4]. Among these frameworks, Deeplearning4j is the most suitable one for integration with Weka [1] because it is also implemented in Java. It supports well-known deep learning architectures such as convolutional neural networks and recurrent neural networks, e.g., long short-term memory networks [5], and training on graphics processing units.

Development of Deeplearning4j has focused on business applications: it is primarily used as an API that enables programmatic integration of deep learning into larger software systems. Our aim with its integration into Weka, which is widely used for research, education and model development through experimentation, is to expand its scope into these areas and provide GUI-based user interaction.

[3] `https://waikato.github.io/weka-wiki/arff_stable/`

[4] `https://www.tensorflow.org/`

3. Related Tools

Software such as Knime [6], Barista [7] and Expresso [8] that exposes deep learning in a GUI use the Python libraries TensorFlow, Keras[5] or Caffe as the backend. The user has to manually set up Python using virtual environments or Anaconda[6] and may have to resolve dependencies of the corresponding deep learning framework. In contrast, our package enables straightforward installation through Weka's GUI by using Java-based components.

4. Software Framework

WekaDeeplearning4j is built as a Weka package and thus makes Deeplearning4j models available in the whole Weka environment. The core models available in the package are: 1) `Dl4jMlpClassifier`, which enables creation of arbitrarily deep feedforward neural networks, including convolutional neural networks, and 2) `RnnSequenceClassifier`, which allows training recurrent neural networks on sequential data such as documents. An appealing feature of our tool, described in Section 4.2, is the support for processing two of the most popular data types in deep learning: images and text.

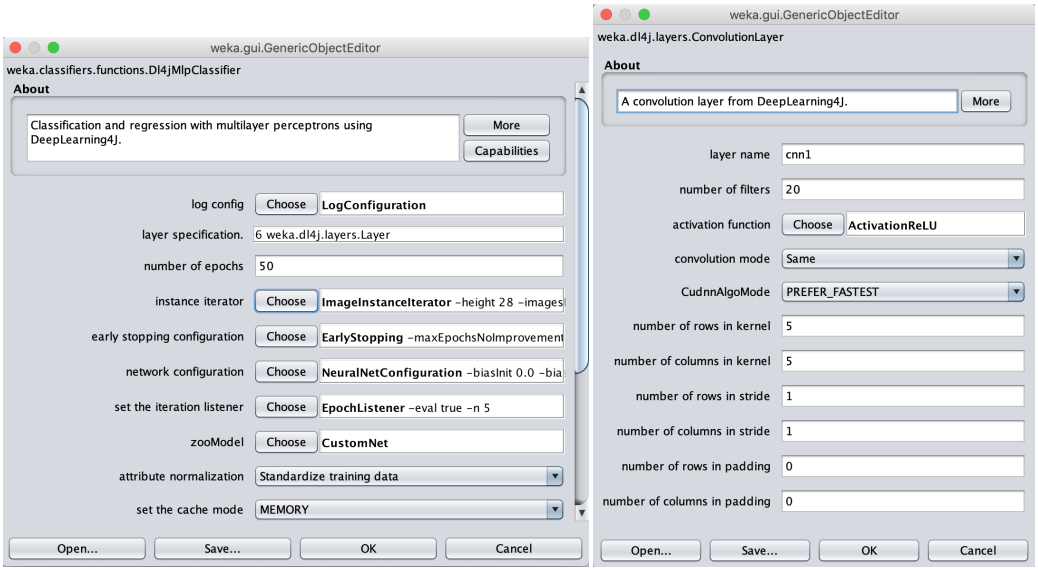
4.1. Network Configuration

The configuration of the `Dl4jMlpClassifier` in the Weka GUI is shown in Figure 1a. The main configuration window exposes a set of basic parameters, such as the number of epochs to train the network, the iterator used to load data (image, textual, tabular or sequence data), an early stopping scheme, and the layer architecture. Fine tuning of configurations can be performed in the `network configuration` window (not shown here). Through this window it is possible to additionally specify network hyper parameters such as the optimization algorithm, the optimization strategy (updater), regularization factors, initialization methods and more. These configurations are globally applied for each layer that is listed in `layer specification`.

The `layer specification` option in the basic model setup window enables the user to create network architectures by stacking different types of neural network layers on top of one another. It is also possible to load a predefined model architecture from the `zooModel` option, which can then be

[5] `https://github.com/fchollet/keras`

[6] `https://anaconda.com/`



(a) (b)

Figure 1: Configuration of the `Dl4jMlpClassifier` model. Figure 1a is the main model configuration window that is shown in the Weka GUI when selecting `Dl4jMlpClassifier`. From here, further nested configurations can be reached, such as the `ConvolutionLayer` setup shown in Figure 1b and more.

further customized through the layer specification. Each element in the layer list can be chosen from the list of available layers and configured with layer specific options. Figure 1b shows the configuration of a `ConvolutionLayer`, giving options for `kernel size`, `stride`, `padding` and more.

The `Dl4jMlpClassifier` can be used for both regression and classification by choosing appropriate loss functions. Additionally, once a classifier has been trained, it is possible to use the serialized `Dl4jMlpClassifier` in the `Dl4jMlpFilter` preprocessing tool for feature extraction (not shown here).

4.2. Data Loading

The package provides so called `InstanceIterators` to load a given dataset in the correct shape. Each iterator enables setting a batch size that determines the size of the mini batches used for training a network.

4.2.1. Image Data

To perform image classification with the package, a dataset must be represented as an ARFF file with one *string* attribute whose values contain the

paths of the image files and another attribute with the corresponding target class values. This file can be loaded as the base dataset in the Weka GUI.

To process this data with deep learning, the `instance iterator` in Figure 1a has to be set to `ImageInstanceIterator`. This iterator additionally specifies the image base directory, the image height and width, and the number of image channels.

4.2.2. Text Data

In deep learning settings, textual data instances (e.g., documents or sentences) are generally represented as sequences of discrete tokens (e.g., words or characters), where each unique token is mapped into a dense vector referred to as an "embedding" [9, 10].

The `RnnTextEmbeddingInstanceIterator` accepts datasets that contain text data and maps each document into an embedding space, based on word embeddings provided by a lookup table through the `location of word vectors` option. This iterator can be used with the `RnnSequenceClassifier` for sequence classification and regression.

Word embeddings can also be generated from the input data with the `Dl4jStringToWord2Vec` and `Dl4jStringToGlove` filters that implement the *Word2Vec* [10] and *Glove* [9] models respectively.

5. Conclusions

We have presented an extension to Weka that enables users to create, train, and test deep neural networks using Weka's GUI, employing Deeplearning4j as the backend. It is targeted at researchers and data science practitioners who want to experiment with deep learning, opening it up to users who prefer GUI-based interaction and want to minimize opportunity cost associated with the setup of software. The extension also allows the incorporation of deep learning models into users' existing Weka workflows.

From a research perspective, our tool provides a unified environment for comparing deep learning models against other machine learning techniques. It also enables application of deep learning in conjunction with many of the "meta" learning schemes implemented in Weka, e.g., cost-sensitive learning, conditional density estimation, and ordinal classification, which opens up opportunities for new research and applications.

Acknowledgements

This work was supported by project 15-UOW-094 of the Marsden Fund of New Zealand. Felipe Bravo-Marquez's work on the text mining aspects of the package was funded by Millennium Institute for Foundational Research on Data.

References

[1] E. Frank, M. A. Hall, I. H. Witten, The WEKA workbench, Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition (2016).

[2] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, NIPS Autodiff Workshop, Long Beach, CA, USA (2017).

[3] F. Seide, A. Agarwal, CNTK: Microsoft's open-source deep-learning toolkit, in: Proc 22nd Int Conf on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2016, p. 2135.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proc 22nd Int Conf on Multimedia, ACM, New York, NY, USA, 2014, pp. 675–678.

[5] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks 61 (2015) 85–117.

[6] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, B. Wiswedel, KNIME: The Konstanz Information Miner, in: Data Analysis, Machine Learning and Applications, Springer, 2008, pp. 319–326.

[7] S. Klemm, A. Scherzinger, D. Drees, X. Jiang, Barista - a graphical tool for designing and training deep neural networks (2018). `arXiv:1802.04626`.

[8] J. H. Dholakiya, R. K. Sarvadevabhatla, R. V. Babu, Expresso: A user-friendly GUI for designing, training and using convolutional neural networks (2015). `arXiv:1505.06605`.

[9] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proc 19th Conf on Empirical Methods in Natural Language Processing, ACL, 2014, pp. 1532–1543.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Neural Information Processing Systems 26, 2013, pp. 3111–3119.

Required Metadata

Nr.	(executable) Software metadata description	Please fill in this column
S1	Current software version	1.5.11
S2	Permanent link to executables of this version	`https://github.com/Waikato/wekaDeeplearning4j/tree/v1.5.11`
S3	Legal Software License	GPLv3
S4	Computing platform/Operating System	Linux, OS X, Microsoft Windows
S5	Installation requirements & dependencies	Java 8 or higher, Weka 3.8.1 or higher
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	`https://deeplearning.cms.waikato.ac.nz/`
S7	Support email for questions	`wekalist@list.scms.waikato.ac.nz`

Table 1: Software metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	1.5.11
C2	Permanent link to code/repository used of this code version	`https://github.com/Waikato/wekaDeeplearning4j`
C3	Legal Code License	GPLv3
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	Java 8, Gradle, Weka
C6	Compilation requirements, operating environments & dependencies	JDK 8 or higher, Weka 3.8.1 or higher
C7	If available Link to developer documentation/manual	`https://waikato.github.io/wekaDeeplearning4j/overview-summary.html`
C8	Support email for questions	`wekalist@list.scms.waikato.ac.nz`

Table 2: Code metadata