MLJFlux: Deep learning interface to the MLJ toolbox

Ayush Shridhar[1] **and Anthony Blaom**[2]

1 International Institute of Information Technology, Bhubaneswar, India **2** Department of Computer Science, University of Auckland

DOI: 10.21105/joss.03375

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Reviewers:

- @krystophny
- @morganericsson

Submitted: 22 April 2021
Published: 15 June 2021

Introduction

We present *MLJFlux.jl* (Shridhar & Bloam, n.d.), an interface between the *MLJ* machine learning toolbox (Blaom et al., 2020) and the *Flux.jl* deep learning framework (Innes, 2018) written in the *Julia* programming language(Bezanson et al., 2017). MLJFlux makes it possible to implement supervised deep learning models while adhering to the MLJ workflow. This means that users familiar with the MLJ design can write their models in Flux with a few slight modifications and perform all tasks provided by the MLJ model spec. The interface also provides options to train the model on different hardware and warm start the model after changing some specific hyper-parameters.

Julia solves the *"two language problem"* in scientific computing, where high-level languages such as Python or Ruby are easy to use but often slow and low-level languages are fast but difficult to use. Using *just-in-time* compilation and *multiple dispatch*, Julia makes the best of both worlds by matching the performance of low-level languages while also being adaptable(*Julia Micro-Benchmarks*, n.d.) .

While there has been significant work towards enabling the creation and delivery of deep learning models with *FastAI.jl* (*Best Practices for Deep Learning in Julia, Inspired by Fastai*, n.d.), the focus has been extensively on neural network paradigms. It provides convenience functions for loading data, modeling and tuning but the process still involves writing the preprocessing pipelines, loss functions, optimizers and other hyper-parameters. MLJFlux.jl tries to remove the need for this boilerplate code by leveraging the MLJ design which makes it ideal for basic prototyping and experimentation.

Statement of Need

While MLJ supports multiple statistical models, it lacks support for any kind of deep learning model. MLJFlux adds support for this by interfacing MLJ with the Flux.jl deep learning framework. Converting a Flux model into an MLJ spec can be done by wrapping it in the appropriate MLJFlux container and specifying other hyper-parameters such as the loss function, optimizer, epochs, and so on. This can now be used like any other MLJ model. MLJFlux models implement the MLJ warm restart interface, which means training can be restarted from where it was left off, when the number of epochs is increased or the optimiser settings (e.g., learning rate) are modified. Consequently, an MLJFlux model can also be wrapped as an MLJ *IteratedModel*, making early stopping, model snapshots, callbacks, cyclic learning rates, and other controls available.

MLJFlux provides four containers that we can enclose our Flux model in. Each model is derived from either `MLJModelInterface.Probabilistic` or `MLJModelInterface.Deterministic` to follow the MLJ design, depending on the type of task. At the core of each wrapper is a *builder* attribute that specifies the neural network architecture (Flux model) given the shape of the data.

MLJFlux has been written with ease of use in mind. The core idea is to allow rapid modeling, tuning and visualization of deep learning models via Flux.jl by reusing the already mature and efficient functionalities offered by MLJ.

References

Best practices for deep learning in julia, inspired by fastai. (n.d.). https://github.com/FluxML/FastAI.jl

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Blaom, A. D., Kiraly, F., Lienart, T., Simillides, Y., Arenas, D., & Vollmer, S. J. (2020). MLJ: A julia package for composable machine learning. *Journal of Open Source Software*, *5*(55), 2704. https://doi.org/10.21105/joss.02704

Innes, M. (2018). Flux: Elegant machine learning with julia. *Journal of Open Source Software*, *3*(25), 602. https://doi.org/10.21105/joss.00602

Julia micro-benchmarks. (n.d.). https://julialang.org/benchmarks/

Shridhar, A., & Bloam, A. (n.d.). *An interface to the deep learning package flux.jl from the MLJ.jl toolbox*. https://github.com/FluxML/MLJFlux.jl