dnadna: a deep learning framework for population genetics inference

Théophile Sanchez, Erik Madison Bray, Pierre Jobic, Jérémy Guez, Anne-Catherine Letournel, Guillaume Charpiat, Jean Cury, Flora Jay

Journal Title Here, 2022, pp. 1–3

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

dnadna: a deep learning framework for population genetics inference

Théophile Sanchez,[1,+] Erik Madison Bray,[1,+] Pierre Jobic,[1,2] Jérémy Guez,[1,3] Anne-Catherine Letournel,[1] Guillaume Charpiat,[1] Jean Cury[1,4,o,*] and Flora Jay[1,o,*]

[1]Université Paris-Saclay, CNRS UMR 9015, INRIA, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France, [2]ENS Paris-Saclay France, [3]UMR7206 Eco-Anthropologie, Muséum National d'Histoire Naturelle, CNRS, Université de Paris, Paris, France, [4]SEED, U1284, INSERM, Université de Paris, Paris, France, [+]These first authors contributed and [o]These last authors contributed equally [*]Corresponding authors. flora.jay@lri.fr jean.cury@normalesup.org
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Motivation: We present `dnadna`, a flexible python-based software for deep learning inference in population genetics. It is task-agnostic and aims at facilitating the development, reproducibility, dissemination, and reusability of neural networks designed for population genetic data.
Results: `dnadna` defines multiple user-friendly workflows. First, users can implement new architectures and tasks, while benefiting from `dnadna` utility functions, training procedure and test environment, which saves time and decreases the likelihood of bugs. Second, the implemented networks can be re-optimized based on user-specified training sets and/or tasks. Newly implemented architectures and pretrained networks are easily shareable with the community for further benchmarking or other applications. Finally, users can apply pretrained networks in order to predict evolutionary history from alternative real or simulated genetic datasets, without requiring extensive knowledge in deep learning or coding in general.
`dnadna` comes with a peer-reviewed, exchangeable neural network, allowing demographic inference from SNP data, that can be used directly or retrained to solve other tasks. Toy networks are also available to ease the exploration of the software, and we expect that the range of available architectures will keep expanding thanks to community contributions.
Availability and Implementation: `dnadna` is a Python ($\geq$3.7) package, its repository is available at `gitlab.com/mlgenetics/dnadna` and its associated documentation at `mlgenetics.gitlab.io/dnadna/`.
Contact and Supplementary Information: flora.jay@lri.fr and jean.cury@normalesup.org

Introduction

In recent years, deep learning has been applied to biology with the hope of facilitating complex data analyses and information discovery, and methods are now flourishing in population genetics (Borowiec *et al.*, 2022). As reviewed by Sanchez *et al.* (2020), we distinguish two families: those processing many summary statistics, with fully connected or convolutional networks and those based on 'raw' genetic data leveraging deep architectures to automatically construct informative features (e.g. Chan *et al.*, 2018; Flagel *et al.*, 2019; Montserrat *et al.*, 2019; Torada *et al.*, 2019; Adrion *et al.*, 2020b; Battey *et al.*, 2020; Sanchez *et al.*, 2020; Battey *et al.*, 2021; Deelder *et al.*, 2021; Fonseca *et al.*, 2021; Gower *et al.*, 2021; Isildak *et al.*, 2021; Wang *et al.*, 2021; Yelmen *et al.*, 2021; Burger *et al.*, 2022; Meisner and Albrechtsen, 2022; Perez *et al.*, 2022; Qin *et al.*, 2022). Previous studies have made their implementations available at least for reproducibility and sometimes with a specific effort for re-usability. Even so, each of them focuses on a specific network for a specific task. Adapting them requires careful understanding of the code and its direct modification since many options are hard-coded. This is not only error-prone, but also leads to a rapid code divergence between parallel projects, accompanied by complex maintenance. The community demands flexible and rigorous tools as demonstrated by `stdpopsim`, a library for population genetic simulations which allows contributions from many researchers (Adrion *et al.*, 2020a). In genomics a suite of tools has been developed to facilitate deep learning applications (e.g. Kopp *et al.*, 2020; Zhang *et al.*, 2021; Routhier *et al.*, 2020). However, none is able to handle population genetic datasets and tasks. For these reasons we developed `dnadna`, Deep Neural Architectures for DNA, a task-agnostic software (in the context of population genetics) that aims at facilitating applications, development, distribution and exchanges around neural networks in the field.

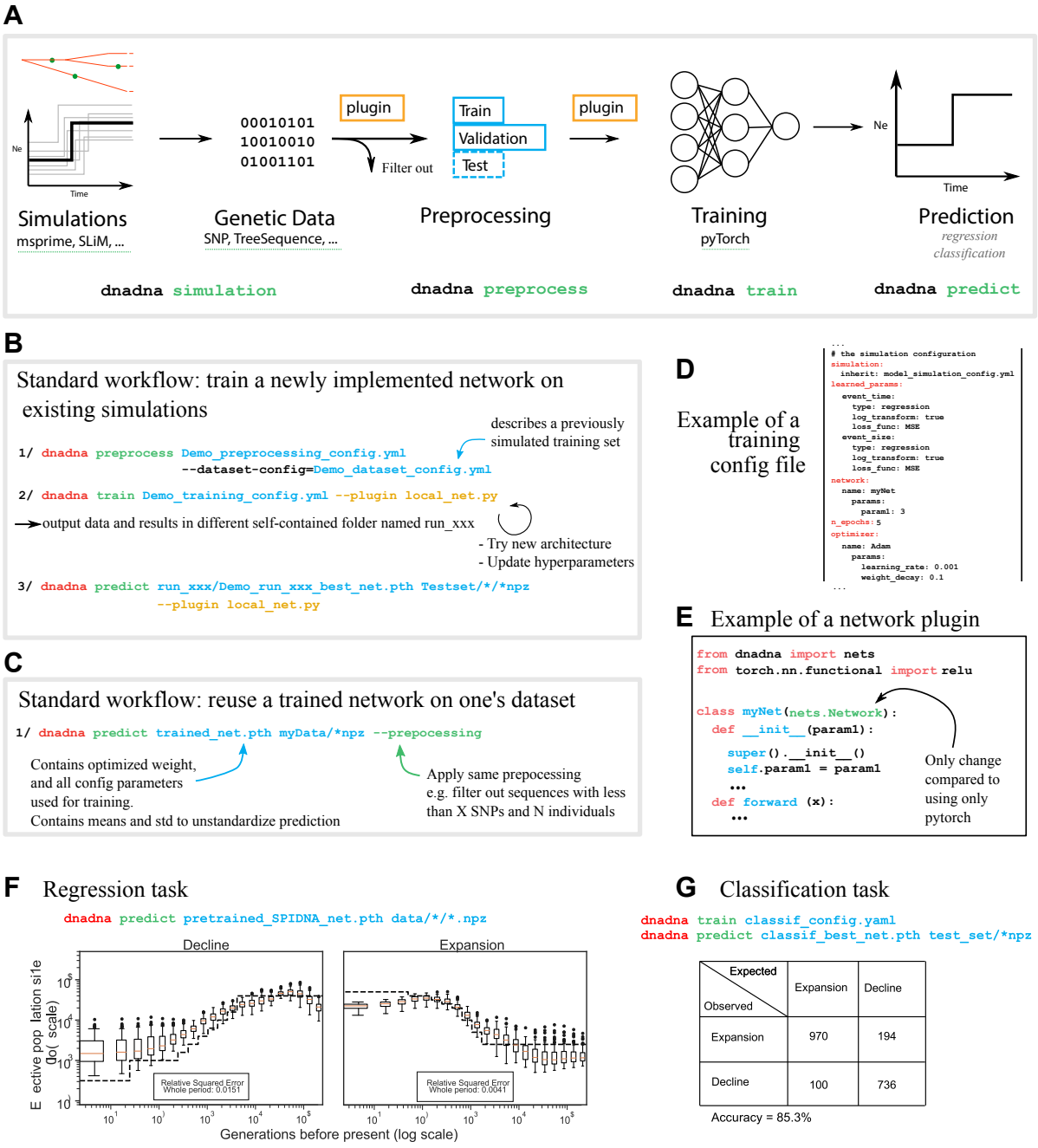

Fig. 1. A: `dnadna` workflow and its corresponding commands. Each step could be done as a standalone: (1) simulation of a large genetic dataset according to evolutionary scenarios and priors; genetic data type is not enforced and can be boolean (classical SNP data), integer (e.g. genotype data 0/1/2), or float (e.g. local density of SNPs as in Gower *et al.* (2021) or summary statistics along the genome as in Xue *et al.* (2021)) ; (2) preprocessing, mainly to filter out examples that do not fit minimal requirements and split the rest into train/validation/test sets; (3) training neural networks; (4) predicting on test or real datasets using optimized neural networks. Note that simulations can be skipped if the user already possesses a labelled dataset. Similarly, training can be skipped if the user reuses a pretrained network. Here is a subset of options at each step: (1a) generating simulations: name of predefined scenario to be simulated and its related parameters, such as number of individuals, number of replicates, mutation rate, demographic parameters; (1b) handling simulations: location on disk and filesystem layout; (2) preprocessing: initial data transformations, filtering values such as minimal number of sampled individuals or SNPs; (3a) architecture design: network name and related arguments (number of filters, layers, ...); (3b) training: loss functions, optimization and training hyperparameters (number of epochs, learning rate, batch size, optimizer name, ...); (3c) on-the-fly data transformations (subsampling, cropping, ...). **B** and **C**: illustration of two standard use cases of `dnadna`. **D**: extract of a training configuration file in YAML format. **E**: view of a plugin python file, that will be passed to `dnadna train`, where users can implement novel networks based on PyTorch. **F**: Illustration of a regression task with a pretrained network. Here we only need to use 'dnadna predict' since the network is already trained. Dotted lines denote true known histories, while boxplots indicate the population sizes predicted for 100 independent genomic regions at each time step. The estimates for a "complete" genome are given by the averaged predicted values. The error corresponds to the relative squared errors averaged over the whole time period. **G**: Illustration of a classification task, following Quickstart tutorial 2. The network was trained on a toy dataset (2000 independent population samples split into training and validation) and tested on 2000 additional samples to discriminate whether the population underwent a decline or an expansion of its size. The contingency table shows that for this classification task, the accuracy is 85.3% on a test set. See `mlgenetics.gitlab.io/dnadna/tutorials.html` for details on both experiments.

Software

`dnadna` is a python-based software for population genetics inference that enables researchers to (1) develop new networks or re-use existing architectures, (2) train them for a given task (regression, classification, or a mix of those), and (3) share them in such a way that users can easily apply these trained networks to their own dataset. In particular, it already implements several neural networks that have been tested for inferring demographic and adaptive histories from genetic data. Pre-trained networks can be used directly on real and simulated genetic data for the prediction step. Networks can also be re-trained on new simulations (e.g. corresponding to another species or evolutionary model) and/or to solve other tasks (e.g. classifying introgressed vs non-introgressed segments or inferring recombination). Finally, any user can implement new architectures and tasks, while benefiting from training procedure, test environment, routines that may be otherwise overlooked (such as proper preprocessing or efficient data loading), and the possibility to easily share a network to facilitate re-use and benchmarking.

Figure 1 provides an overview of `dnadna` steps. It is accessible without coding knowledge thanks to its YAML configuration files which provide the user with a variety of options at each step of the process. Because each use-case has its own specificity, we developed a system which allows users to implement plugins (e.g. data transformations, simulators, or networks) without modifying the core of the code.

`dnadna` is a Python ($\geq$3.7) package with multiple dependencies, the main one being the open source machine learning library PyTorch. It has a command line interface and an API (`mlgenetics.gitlab.io/dnadna/api.html`). It is highly flexible thanks to a structured configuration file system based on YAML and JSON Schema. `dnadna` is dual-licensed under the GNU Lesser General Public License (LGPLv3+) and the compatible CeCILL-C Free Software License Agreement. Release 1.0 is available from PyPI at `pypi.org/project/dnadna/` and Anaconda at `anaconda.org/mlgenetics/dnadna`. Docker images are available at `hub.docker.com/u/mlgenetics`.

Tutorial examples

We showcase various `dnadna` use-cases via tutorials that will continue to expand. It is not required to have coding knowledge to perform similar tasks. A first tutorial walks the user through the complete process from configuring and generating simulated genetic data, to running data pre-processing, and training a convolutional network to solve a regression task (here predicting the parameters of a two-step population size history). A second tutorial solves a classification task instead (Figure 1G). A third tutorial helps users who already have simulated data, to familiarize themselves with `dnadna` and train a SPIDNA network on a provided dataset. Finally, we provide tutorials in the form of jupyter notebooks (`mlgenetics.gitlab.io/dnadna/tutorials.html` and Figure 1F).

Acknowledgements

TAU and Kepler GPU platforms. DIM One Health 2017 RPH17094JJP, Human Frontier Science Project RGY0075/2019, Paris-Saclay CDS 2.0 (IRS), CNRS-80Prime TransIA, ANR-20-CE45-0010-01 RoDAPoG for funding. M Fumagalli and the participants of the School "Inference using full genome data" (DFG SPP1819) for beta-testing. S Ribeiro for her comments.

References

Adrion, J. R. *et al.* (2020a). A community-maintained standard library of population genetic models. *eLife*, **9**, e54967.

Adrion, J. R. *et al.* (2020b). Predicting the Landscape of Recombination Using Deep Learning. *Molecular Biology and Evolution*, **37**(6), 1790–1808.

Battey, C. *et al.* (2020). Predicting geographic location from genetic variation with deep neural networks. *eLife*, **9**, e54507.

Battey, C. J. *et al.* (2021). Visualizing population structure with variational autoencoders. *G3*, **11**(1), 1–11.

Borowiec, M. L. *et al.* (2022). Deep learning as a tool for ecology and evolution. *Methods in Ecology and Evolution*, **13**(8), 1640–1660.

Burger, K. *et al.* (2022). Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown. *PLOS Computational Biology*, **18**, e1010407.

Chan, J. *et al.* (2018). A Likelihood-Free Inference Framework for Population Genetic Data using Exchangeable Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31.

Deelder, W. *et al.* (2021). Using deep learning to identify recent positive selection in malaria parasite sequence data. *Malaria Journal*, **20**(1), 270.

Flagel, L. *et al.* (2019). The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference. *Molecular Biology and Evolution*, **36**(2), 220–238.

Fonseca, E. M. *et al.* (2021). Phylogeographic model selection using convolutional neural networks. *Molecular Ecology Resources*, **21**(8), 2661–2675.

Gower, G. *et al.* (2021). Detecting adaptive introgression in human evolution using convolutional neural networks. *eLife*, **10**, e64669.

Isildak, U. *et al.* (2021). Distinguishing between recent balancing selection and incomplete sweep using deep neural networks. *Molecular Ecology Resources*, **21**(8), 2706–2718.

Kopp, W. *et al.* (2020). Deep learning for genomics using Janggu. *Nature Communications*, **11**(1), 3488.

Meisner, J. and Albrechtsen, A. (2022). Haplotype and population structure inference using neural networks in whole-genome sequencing data. *Genome Research*, **32**(8), 1542–1552.

Montserrat, D. M. *et al.* (2019). Class-Conditional VAE-GAN for Local-Ancestry Simulation. *arXiv:1911.13220*.

Perez, M. F. *et al.* (2022). Coalescent-based species delimitation meets deep learning: Insights from a highly fragmented cactus system. *Molecular Ecology Resources*, **22**(3), 1016–1028.

Qin, X. *et al.* (2022). Deciphering signatures of natural selection via deep learning. *Briefings in Bioinformatics*, **23**(5), bbac354.

Routhier, E. *et al.* (2020). Keras_dna: A Wrapper For Fast Implementation Of Deep Learning Models In Genomics. *Bioinformatics*, **37**(11), 1593–1594.

Sanchez, T. *et al.* (2020). Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation. *Molecular Ecology Resources*, **21**(8), 2645–2660.

Torada, L. *et al.* (2019). ImaGene: a convolutional neural network to quantify natural selection from genomic data. *BMC Bioinformatics*, **20**(S9), 337.

Wang, Z. *et al.* (2021). Automatic inference of demographic parameters using generative adversarial networks. *Molecular Ecology Resources*, **21**(8).

Xue, A. T. *et al.* (2021). Discovery of Ongoing Selective Sweeps within Anopheles Mosquito Populations Using Deep Learning. *Molecular Biology and Evolution*, **38**(3), 1168–1183.

Yelmen, B. *et al.* (2021). Creating artificial human genomes using generative neural networks. *PLOS Genetics*, **17**(2), e1009303.

Zhang, Z. *et al.* (2021). An automated framework for efficiently designing deep convolutional neural networks in genomics. *Nature Machine Intelligence*, **3**(5), 392–400.