# DLOPT: Deep Learning Optimization Library

**Andrés Camero**[1,*]**, Jamal Toutouh**[1]**, and Enrique Alba**[1]

[1]Universidad de Málaga, Departamento de Lenguajes y Ciencias de la Computación, Málaga, España
*andrescamero@uma.es

## ABSTRACT

Deep learning hyper-parameter optimization is a tough task. Finding an appropriate network configuration is a key to success, however most of the times this labor is roughly done. In this work we introduce a novel library to tackle this problem, the Deep Learning Optimization Library: DLOPT. We briefly describe its architecture and present a set of use examples. This is an open source project developed under the GNU GPL v3 license and it is freely available at https://github.com/acamero/dlopt.

**Keywords:** deep learning, deep neuroevolution, optimization

## 1 Introduction

Hyper-parameter optimization is a challenging task, specially in the context of deep neural networks (DNN) and deep learning (DL). Its main aim is to find/select the most suitable DNN configuration parameters for a specific problem[1,2]. These parameters determine the activation functions, the number of hidden layers, the kernel size of a layer, etc.

The number of parameters and the large number of their possible values define a high-dimensional search space. Even though it is hard to find a "good" configuration, most DL studies in the literature rely on expert knowledge and on manual exploration strategies to address this problem[3]. Therefore, it seems mandatory to use an automatic intelligent tool to find an efficient hyper-parameter configuration (e.g., grid, evolutionary or random search)[4–6].

In this context, *evolutionary algorithms* (EAs)[7] emerged as efficient stochastic methods to address hyper-parameter optimization problems. Indeed, these algorithms have been used in the literature to address this problem in shallower neural networks[8,9]. However, these solutions are not appropriate in a DL context due to the high complexity of DNNs[10].

Nowadays, some new studies have analyzed the use of different variants of EAs to provide efficient DNN configurations, giving rise to *deep neuroevolution*[11–14]. These studies have shown that EAs are adequate to deal with DNN hyper-parameter configuration problem and that they require less computational resources than traditional approaches (e.g. backpropagation, grid search). Thus, these competitive results motivated us to develop a software library, the *Deep Learning Optimization Library (DLOPT)*, to ease the use of these promising algorithms to address hyper-parameter optimization without requiring a thorough knowledge in EAs.

In this report we briefly introduce DLOP, a library for deep learning hyper-parameter optimization. First, we outline its architecture. Then, Section 3 presents some examples and summarizes their main results. Finally, Section 4 presents the essential conclusions and formulates the principal lines for future work.

## 2 Core Architecture

The core of DLOPT is composed of three classes: `ModelOptimization`, `Problem`, and `Solution`. This division aims to decouple the problem being solved (e.g. optimizing the architecture of a neural network) and the technique used to solve the problem (e.g. using a genetic algorithm to optimize the testing error of a neural network). At a glance, the `ModelOptimization` is an abstract class that is the basis for solving the hyper-parameter optimization problem. The `Problem` class is an abstract definition that acts as the basis for encoding, decoding and evaluating a solution. And finally, a `Solution` is a class that encapsulates an artificial neural network encoded by a specific problem, as well as the metrics (values of the fitness) calculated.

The library presents multiple implementations of the referred classes, as well as many utilities (e.g. *random samplings*[15,16], recurrent neural network builders and trainers, and data loader, among others), and we are still working on new state-of-the-art techniques to tackle the hyper-parametrization optimization problem.

The implementations (of the core classes) are divided into two major categories: architecture and weight. Both problems are related to each other, however we decided to split the code to ease the usability. The *architecture* submodule intends to solve the problem of optimizing the architecture of a given network, while the training (i.e. the optimization of the weights) is

done by commonly used techniques (e.g. backpropagation-based methods). On the other hand, the *weight* submodule focuses on solving the optimization of the synaptic weights. The library also includes a submodule named *tools*. This module offers a basic command line interface to use some functionalities. This interface can be broaden by extending the abstract class `ActionBase`.

DLOPT is implemented on Python 3 and it relies on Keras (version 2.1) and Tensorflow (version 1.3)[17]. The source code is available under the GNU GPL v3 license (https://github.com/acamero/dlopt).

## 3 DLOPT in Action

This section aims at illustrating the usage and the performance of DLOPT. In this case we focused on the use of our library to deal with a specific type of DNNs, recurrent neural networks (RNN). Thus, we tackled a well-known test problem in the domain of DL by using RNNs, the *sine wave*. Despite its simplicity, we selected sine wave because by adding this type of waves it is possible to approximate any periodic waveform[18].

In order to ease the reproductivity of the presented experimentation the library includes an executable file to run these examples. This file is located in the following path: `examples/sin/run.sh` (see Listing 1). Two different examples are implemented to be run: the *MAE random sampling*[15,16] (Line 2) and the hyper-parameter optimization of a RNN by using the *MAE random sampling* (Line 3), in order to introduce the main features of our library.

**Listing 1.** Shell script implemented in `run.sh`.

```
1  #!/bin/bash
2  python ../../dlopt/tools/main.py --config mae-rand-samp-sin.json --verbose=1
3  python ../../dlopt/tools/main.py --config mae-optimization-sin.json --verbose=1
```

Listing 2 summarizes the results provided by the MAE random sampling of a given architecture. In this case, the RNN applies a look back of two (Line 1) and its architecture is defined by one neuron in the input layer, two hidden layers with two neurons each, and one neuron output layer (see Line 2). The main results of the sampling are shown in lines 3 to 9.

**Listing 2.** MAE random sampling results of a given architecture.

```
1  "look_back": 2,
2  "architecture": [1, 2, 2, 1],
3  "metrics": {
4          "log_p": -29.906005813483684,
5          "p": 1.0279848149134584e-13,
6          "mean": 0.81072833698353519,
7          "samples": [0.61528207663056766, ..., 0.72354892475102595],
8          "std": 0.1104613706995357
9  }
```

Listing 3 shows a portion of the results provided by the evolutionary hyper-parameter optimization example. The optimization algorithm returns neural network model (configuration) that can be easily loaded by using Keras interface, and therefore, it can be used to address any DL problem.

**Listing 3.** Evolutionary hyper-parameter optimization results.

```
1  "fitness": { 'log_p': -12.215031852558125},
2  "layers": [1, 12, 13, 9, 10, 12, 6, 1],
3  "look_back": 17,
4  "config": [{
5    "class_name": "LSTM",
6    "config": {
7          "activation": "tanh",
8          "units": 12,
9  ...
```

In this case, the resulted RNN hyper-parameter configuration to address sine wave problem resulted with a fitness value of -12.215 (Line 1). The parameters presented in Listing 3 define a network with a look back of 17 (Line 3) and an architecture with six hidden layers with 12, 13, 9, 10, 12, and 6 neurons, respectively (Line 2). From Line 4 upwards, the output specifies

the configuration of each hidden layer. In the example in Listing 3, the first hidden layer includes 12 LSTM neurons (Line 5) that apply *tanh* as activation function (Line 7).

## 4 Conclusions

In this work we introduce DLOPT, a library for deep learning hyper-parameter optimization. We briefly present its core architecture and a set of use examples.

Despite being a work in progress, DLOPT is fully operational. Up to date, the library has been used solely for academic purposes, however we plan to transfer it to the industry. Most of the hyper-parameter optimization techniques included in DLOPT are originals and are publicly available, i.e. the source code and several scientific publications[11,15,16]. Therefore, we consider that our library presents enough evidence to be considered as a useful and valid tool for hyper-parameter optimization.

The design of DLOPT is intended to be flexible enough to be extended and adapted to a wide variety of hyper-parameter problems, including architecture and weight optimizations. Hence, as future work we propose to extend the library by adding new custom designed optimization algorithms and to add new problems.

## References

1. Bergstra, J. S., Bardenet, R., Bengio, Y. & Kégl, B. Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. & Weinberger, K. Q. (eds.) *Advances in Neural Information Processing Systems 24*, 2546–2554 (Curran Associates, Inc., 2011).

2. Jozefowicz, R., Zaremba, W. & Sutskever, I. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 2342–2350 (JMLR.org, 2015).

3. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436 (2015).

4. Albelwi, S. & Mahmood, A. A framework for designing the architectures of deep convolutional neural networks. *Entropy* **19**, 242 (2017).

5. Domhan, T., Springenberg, J. T. & Hutter, F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, 3460–3468 (AAAI Press, 2015).

6. Smithson, S. C., Yang, G., Gross, W. J. & Meyer, B. H. Neural networks designing neural networks: multi-objective hyper-parameter optimization. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*, 1–8 (IEEE, 2016).

7. Back, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford university press, 1996).

8. Alba, E. & Martí, R. *Metaheuristic Procedures for Training Neural Networks*, vol. 35 (Springer Science & Business Media, 2006).

9. Yao, X. Evolving artificial neural networks. *Proc. IEEE* **87**, 1423–1447 (1999). DOI 10.1109/5.784219.

10. Ojha, V. K., Abraham, A. & Snášel, V. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Eng. Appl. Artif. Intell.* **60**, 97–116 (2017).

11. Camero, A., Toutouh, J., Stolfi, D. H. & Alba, E. Evolutionary deep learning for car park occupancy prediction in smart cities. In *Learning and Intelligent OptimizatioN (LION) 12*, 1–15 (Springer, 2018).

12. Miikkulainen, R. *et al.* Evolving Deep Neural Networks. *arXiv preprint arXiv:1703.00548* (2017). `1703.00548`.

13. Morse, G. & Stanley, K. O. Simple Evolutionary Optimization Can Rival Stochastic Gradient Descent in Neural Networks. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, 477–484 (ACM, 2016). DOI 10.1145/2908812.2908916.

14. Such, F. P. *et al.* Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv preprint arXiv:1712.06567* (2017). `1712.06567`.

15. Camero, A., Toutouh, J. & Alba, E. Low-cost recurrent neural network expected performance evaluation. *arXiv preprint arXiv:1805.07159* (2018).

16. Camero, A., Toutouh, J. & Alba, E. Comparing deep recurrent networks based on the mae random sampling, a first approach. In *Conference of the Spanish Association for Artificial Intelligence (CAEPIA) 2018*, 1–10 (Springer, 2018).

17. Abadi, M. *et al.* Tensorflow: A system for large-scale machine learning. In *OSDI*, vol. 16, 265–283 (2016).

18. Bracewell, R. N. & Bracewell, R. N. *The Fourier transform and its applications*, vol. 31999 (McGraw-Hill New York, 1986).

## Acknowledgements

## Author contributions statement

Conceptualization, A.C. and J.T.; Software, A.C.; Validation, A.C. and J.T.; Investigation, A.C. and J.T.; Resources, E.A.; Writing–Original Draft Preparation, A.C. and J.T.; Writing–Review & Editing, A.C., J.T. and E.A.; Supervision, E.A.; Funding Acquisition, E.A.