Software Impacts 21 (2024) 100675

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

deforce: Derivative-free algorithms for optimizing Cascade Forward Neural Networks

Nguyen Van Thieu [a,*], Hoang Nguyen [b,c], Harish Garg [d,*], Gia Sirbiladze [e]

[a] Faculty of Computer Science, PHENIKAA University, Yen Nghia, Ha Dong, Hanoi, 12116, Viet Nam
[b] Department of Surface Mining, Mining Faculty, Hanoi University of Mining and Geology, Bac Tu Liem District, 18 Vien Str., Duc Thang Ward, Hanoi, 100000, Viet Nam
[c] Innovations for Sustainable and Responsible Mining (ISRM) Research Group, Hanoi University of Mining and Geology, Bac Tu Liem District, 18 Vien Str., Duc Thang Ward, Hanoi, 100000, Viet Nam
[d] Department of Mathematics, Thapar Institute of Engineering and Technology (Deemed University), Patiala 147004, Punjab, India
[e] Department of Computer Sciences, Faculty of Exact and Natural Sciences, Ivane Javakhishvili Tbilisi State University, University St. 13, 0186 Tbilisi, Georgia

ARTICLE INFO

Keywords:
Cascade Forward Neural Network
Derivative free optimization
Hybrid model
Open-source software
Machine learning
Soft computing

ABSTRACT

This paper aims to introduce the 'deforce' framework, an open-source Python library constituted on top of Numpy, Scikit-Learn, PyTorch, and Mealpy. This framework provides hybrid models that combine derivative-free techniques with Cascade Forward Neural Networks (CFNNs). By inheriting from scikit-learn's estimator, deforce's models ensure easy integration into existing machine learning pipelines. It also has many advantages, including a simple installation process, a user-friendly interface, and adaptability to various user requirements. For researchers and practitioners looking to improve CFNN performance with minimal implementation effort, deforce offers a useful and approachable option.

Code metadata

Current code version	*v1.0.0*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2024-106
Permanent link to reproducible capsule	
Legal code license	*GNU General Public License (GPL) V3*
Code versioning system used	*Git*
Software code languages, tools and services used	*Python*
Compilation requirements, operating environments and dependencies	*numpy, scipy, scikit-learn, pandas, mealpy, permetrics, torch, skorch*
If available, link to developer documentation/manual	https://deforce.readthedocs.io/
Support email for questions	nguyenthieu2102@gmail.com

1. Motivation and overview

One particular type of neural network that usually has one hidden layer is called a Cascade Forward Neural Network (CFNN) [1]. It differs from other types of neural networks, such as multi-layer perceptron's and feed-forward neural networks, since it has additional direct connections from the input layer to the output layer [2]. This extra link enables the network to capture any linear correlations between the input and output. Besides, it has potential to accelerate learning and improve the network's ability to generalize from input to output [3]. CFNN has been used in a variety of applications, including rock mining process selection [4], ambient air temperature prediction [5], time series prediction (such as financial forecasting and weather prediction), and other time-dependent data analysis [1,6]. These applications show the CFNN's capacity to handle complex datasets and make accurate predictions, making it a useful tool in a variety of scientific and engineering fields.

While Cascade Forward Neural Networks (CFNNs) offer several advantages in terms of design and learning capabilities, they also have certain drawbacks, particularly with their training method which is Gradient descent algorithm. CFNNs might get stuck in local minima while training when Gradient descent algorithm converges to

DOI of original article: https://doi.org/10.1016/j.resourpol.2021.102300.
* Corresponding authors.
E-mail addresses: thieu.nguyenvan@phenikaa-uni.edu.vn (N. Van Thieu), nguyenhoang@humg.edu.vn (H. Nguyen), harishg58iitr@gmail.com, harish.garg@thapar.edu (H. Garg), gia.sirbiladze@tsu.ge (G. Sirbiladze).

https://doi.org/10.1016/j.simpa.2024.100675
Received 29 April 2024; Received in revised form 28 May 2024; Accepted 12 June 2024

a position other than the error surface's global minimum, resulting in inferior performance [7]. The network may undergo vanishing or exploding gradients, particularly in networks with many layers or complex structures [8]. CFNNs are more difficult to implement than regular multilayer perceptrons. This complexity results from the increased direct connections between layers, which must be carefully managed during the training process [2]. The direct connections between the input and all subsequent layers can result in higher computing intensity, leading to longer training times and the demand of extra computational resources [9]. To efficiently train CFNNs, more complicated optimization techniques may be required. Standard optimization approaches may be less effective due to the cascade of connections. While CFNNs can capture complicated patterns, there is a risk of overfitting the training data, which could limit the model's capacity to generalize to new, unseen data [10]. Furthermore, the selection of hyperparameters has an impact on CFNN performance. Finding the best setup can be a difficult and time-consuming task [11].

Recently, a class of methods called derivative-free optimization (DFO) has been introduced for the purpose of optimizing neural networks. These are gradient-free optimization techniques which do not require gradient knowledge and hence they are beneficial when derivatives are not accessible or cannot be computed easily [12]. They prove particularly valuable for black-box optimization situations where the objective function is represented by a simulation or oracle that does not supply derivative information [13]. Examples include direct search approaches [14], model-based strategies [15], and global optimization algorithms [16]. The most widely used of these are global optimization algorithms, particularly metaheuristic algorithm [17]. They can be adopted to improve CFNNs in the following ways:

- Network Training (Weights Optimization): Metaheuristics can determine the best set of weights for a CFNN by more effectively navigating the search space than traditional approaches. They can avoid local minima, a major problem with gradient descent algorithms, by using exploration schemes within the search space [18]. This could improve the network's generalizability and overall performance.
- Hyperparameter Optimization: The efficacy of CFNNs is strongly dependent on hyperparameter selection, which includes variables such as the number of neurons, learning rate, and activation functions. Metaheuristics can automate hyperparameter tweaking by successfully scanning the hyperparameter space to find the best combination for performance [19]. This may considerably minimize the time and effort required for manual tuning.

Derivative-free optimization (DFO) offers several benefits, including the absence of the necessity for gradient information. This makes it ideal for optimizing CFNNs when the gradient is difficult to compute or not available. DFOs can optimize complex CFNN landscapes by avoiding local minima and finding global optima [20]. They are flexible and can handle a variety of optimization problems, including discontinuous, non-differentiable, and multimodal functions [21]. Additionally, many DFO algorithms can operate in parallel [22], allowing for expedited computation by distributing the search process among multiple processors. However, DFO algorithms have various limitations, such as: DFOs can be more computationally expensive than gradient-based approaches, particularly for high-dimensional problems, making CFNN optimization time-consuming. DFOs' convergence rate can be slower than gradient-based approaches, resulting in longer training times for CFNNs [23]. DFOs frequently have numerous hyperparameters that must be adjusted. Finding the proper settings can be a difficult undertaking that requires significant testing and fine-tuning [24]. As the size and complexity of the CFNN increase, DFOs' efficiency in identifying high-quality solutions may decline, providing scalability issues for the optimization process [25]. In overall, while DFOs provide a viable alternative to standard gradient-based optimization approaches for CFNNs, particularly in scenarios when gradient information is lacking or unfounded, they also pose computational resource and efficiency difficulties. The choice between DFOs and gradient-based approaches for optimizing CFNNs should be made depending on the problem's specific requirements and constraints.

To the best of the author knowledge, there is currently no framework that offers CFNNs (Cascade feedforward neural networks) models and hybrid models that combine DFO (Derivative-free optimization) approaches with CFNNs, despite the facts that they are frequently utilized and have been around for a long time. In this paper, we propose a framework named ***'deforce'*** which offers both standard CFNN models (trained by Gradient descent algorithms) and hybrid CFNN models (trained by DFO methods). Moreover, we introduce an autonomous hyperparameter adjustment mechanism for CFNNs. By adding four additional classes (predictive models): CfnRegressor, CfnClassifier, DfoCfnRegressor, and DfoCfnClassifier to the Scikit-Learn package, deforce expands its capabilities. These models improve learning capacity and predictive accuracy, making them very helpful in regression and classification problems. They serve as beneficial tools for scholars and data scientists that want to explore data more using CFNN-based models.

2. Software structure

Based on Python's OOP architecture, classes, and modules, we designed and constructed the compact structure of the proposed deforce library, as shown Fig. 1. The figure shows the fundamental components of modules, packages, and classes that make up proposed estimators (predictive models). In this form, a rhombus shape indicates a package that contains rectangle-shaped sub-modules. Within these sub-modules, proposed classes are indicated by ellipses.

The toolkit package includes several important modules: The activators module provides activation functions for CFNN networks. The scalers module defines methods and classes for data scaling. The validators module handles parameter validation and evaluation for model inputs. The preprocessor modules perform data preparation functions, besides, it contains a Data class that can be used for reading and loading data. The metrics module provides evaluation metrics taken from the PerMetrics [26] library. These modules provide important auxiliary functions that will be utilized throughout the library.

The model package consists of five core modules: base_cfn_torch and gd_cfn, which provide CfnRegressor and CfnClassifier classes for regression and classification using Gradient-based CFNN models. These modules are created on top of the PyTorch and Skorch libraries to take advantage of PyTorch's Gradient-based optimizers. base_cfn_numpy and dfo_cfn are intended to provide DfoCfnRegressor and DfoCfnClassifier classes, respectively, which provide hybrid models for regression and classification issues combining derivative-free optimizers and CFNN networks. These modules are built on the Numpy and Mealpy libraries, and they make use of Mealpy's Derivative Free Algorithms. Finally, the dfo_tune_cfn module features the DfoTuneCfn class, which allows users to tune the parameters of standard CFNN networks using derivative-free techniques.

This hierarchical design seeks to encapsulate capabilities into reusable modules and classes, leveraging existing libraries to provide efficient solutions for a variety of machine learning applications within the deforce library.

3. Software functionalities

As stated above, this library provides standard CFNN models (gradient-based models). Furthermore, we offer DFO methods combined with CFNNs for two distinct purposes, namely: network training (weights optimization) and hyperparameter optimization. As a result, in this section, we will discuss the key features of three types of models: standard neural networks, hybrid neural networks with DFO for weight

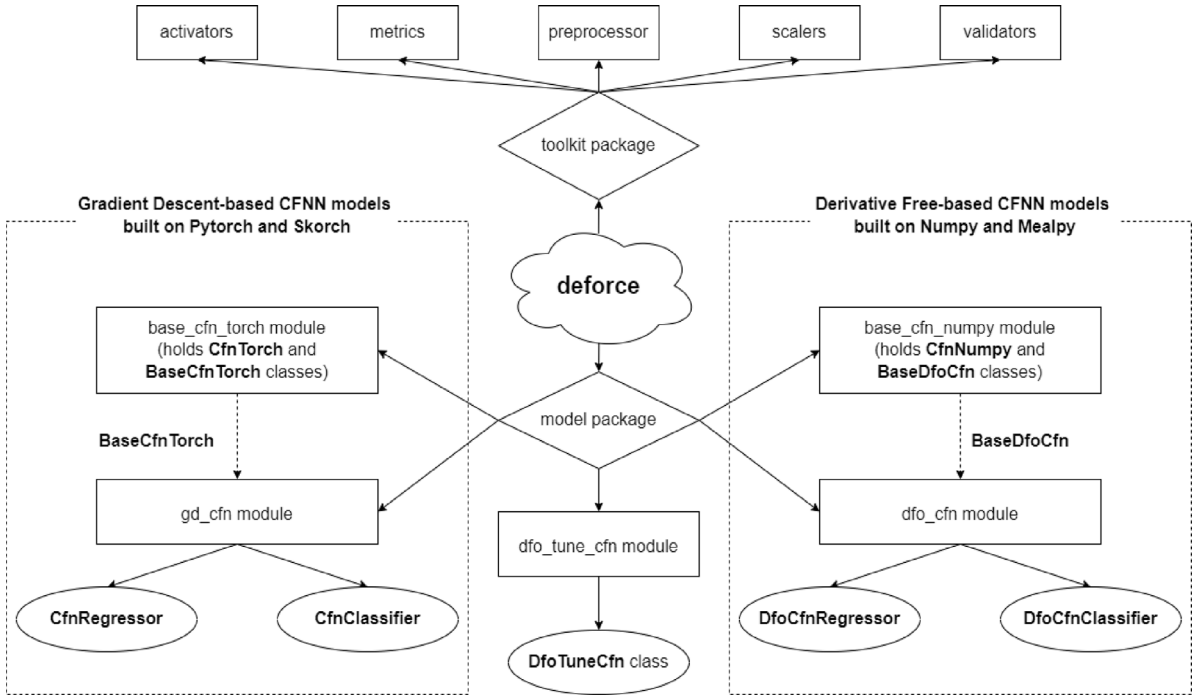

Fig. 1. The structure overview of 'deforce' framework.

optimization, and hybrid neural networks with DFO for hyperparameter optimization.

It is important to note that, as mentioned above, the main classes in this library inherit from the scikit-learn Estimator class. Therefore, they all include the necessary functions, such as fit() for training the model and predict() for predicting values. As a result, the following code examples will only show how users can define a model type. The way to use the model is the same as any other algorithm in the scikit-learn library. Users can apply all models from the "deforce" library to the data they have. The only requirement is that their data must be in supervised learning format, meaning it has features and labels. This is our biggest advantage, a library that defines a general CFNN-based models that can be applied to different regression and classification problems.

3.1. Standard CFNN models (Gradient descent-based models)

To use traditional CFNN models, users can import two classes, CfnRegressor and CfnClassifier, and then initialize the corresponding object. Listing 1 illustrates the syntax for importing and initializing the object. The most notable parameter is 'optimizer', which is used to select various gradient descent-based algorithms available from the PyTorch library,[1] such as Adam optimizer, Adamax optimizer, RMSprop optimizer, etc.

3.2. Hybrid weight-optimized DFO-CFNN models

The hybrid models, in which derivate free optimization (DFO) techniques are used instead of gradient descent procedures to train the CFNN models, are referred to as "hybrid weight-optimized DFO-CFNN models". Similar to the standard CFNN models, users need to import two classes, DfoCfnRegressor and DfoCfnClassifier. Listing 2 presents the code snippet to achieve this. It is worth mentioning that the 'optimizer' parameter reflects the DFO algorithm available from the Mealpy library,[2] such as genetic algorithm, whale optimization, particle swarm optimization, and so on. As can be seen, although these are different models, the parameter settings are quite similar and straightforward. All the important logic has been implemented internally to make the simplest possible interface for users.

3.3. Hybrid hyperparameter-optimized DFO-CFNN models

In this scenario, we refer to these hybrid models as "hybrid hyperparameter-optimized DFO-CFNN models" since the parameters of CFNN models are optimized using Derivative free optimization (DFO) techniques. Gradient descent algorithms are still used to train the CFNN model during the parameter optimization process. Users have to invoke a class named DfoTuneCfn, as shown in Listing 3. It is vital to note that users must use variables from the Mealpy library, such as StringVar, IntegerVar, and MixedSetVar, to specify boundaries for CFNN model hyperparameters such as hidden_size, act1_name, act2_name, max_epochs, batch_size, optimizer, etc. These bounds represent the possible values for the parameters, and the DFO algorithm's task is to decide which set of values produces the best-performing model.

4. Impact of proposed software

To the best of our knowledge, the 'deforce' library is the first open-source library that provides both standard CFNN models (gradient-based models) and hybrid CFNN models (derivative-free models). Our proposed 'deforce' library can be used to address the research questions listed below.

- Optimization of neural network architectures: How can hybrid models that combine derivative-free optimization approaches with Cascade Forward Neural Networks (CFNNs) improve classic gradient-based methods?
- Performance improvement in regression and classification: What are the advantages of adopting hybrid CFNN models in regression and classification applications over traditional machine learning models? How do hybrid CFNN models perform on various benchmark datasets and metrics?

[1] https://pytorch.org/docs/stable/optim.html
[2] https://github.com/thieu1995/mealpy

```
from deforce import CfnRegressor, CfnClassifier

model = CfnRegressor(hidden_size=20, act1_name="relu", act2_name="tanh",
                     obj_name="MSE", max_epochs=500, batch_size=4,
                     optimizer="SGD", optimizer_paras=None,
                     verbose=True, seed=42)

model = CfnClassifier(hidden_size=50, act1_name="elu", act2_name="sigmoid",
                      obj_name="BCEL", max_epochs=500, batch_size=16,
                      optimizer="SGD", optimizer_paras=None,
                      verbose=True, seed=42)
```

Listing 1: Snippet code to import and define the traditional CFNN models

```
from deforce import DfoCfnRegressor, DfoCfnClassifier

opt_paras = {"name": "GA", "epoch": 250, "pop_size": 30}
model = DfoCfnRegressor(hidden_size=10, obj_name="MSE",
                        act1_name="tanh", act2_name="sigmoid",
                        optimizer="BaseGA", optimizer_paras=opt_paras,
                        verbose=True, seed=42)

opt_paras = {"name": "WOA", "epoch": 100, "pop_size": 30}
model = DfoCfnClassifier(hidden_size=20, obj_name="NPV",
                         act1_name="tanh", act2_name="sigmoid",
                         optimizer="OriginalWOA", optimizer_paras=opt_paras,
                         verbose=True, seed=42)
```

Listing 2: Snippet code to import and define the hybrid weight-optimized DFO-CFNN models

- Scalability and benchmarking: How scalable are hybrid CFNN models for large-scale datasets? What benchmarks may be set up using the 'deforce' package to standardize the evaluation of CFNN-based models?
- Application-specific predictive modeling: How might hybrid CFNN models be used in mining, construction, bioinformatics, and astrophysics to increase prediction accuracy and robustness? What specific benefits do hybrid CFNN models bring to predictive maintenance, resource allocation, and decision-making processes?
- Comparative analysis of optimization algorithms: How do different derivative-free optimization techniques perform when used to train CFNN models? Which metaheuristic algorithms perform better on various datasets and problem domains?
- Adaptability and flexibility of machine learning solutions: How can the 'deforce' library's customization and flexibility be used to provide flexible machine learning solutions adapted to specific research or industry requirements? What are the practical problems and solutions for incorporating 'Deforce' into existing machine learning pipelines?
- Cross-disciplinary applications: How might hybrid CFNN models be used to extract new insights from complicated datasets in interdisciplinary fields of research like social sciences and environmental studies? How effective are hybrid CFNN models for assessing and predicting behaviors in non-traditional machine learning domains?

In summary, the proposed library has the potential to alter the machine learning community by improving research capacities, encouraging innovation, and facilitating the practical deployment of CFNN-based models across a wide range of fields.

In our research endeavor, deforce plays a significant role in the utilization and construction of CFNN-based models to generate efficient predictive models. Two exemplary cases depicted in the research documents are supported by the deforce library.

- The research investigates the relationship between production parameters, ore grades, and mine life when estimating mining capital costs (MCC) for open pit mining projects [7].
- The research discusses the difficulty of working with soft soils, particularly clay layers, during building. Soft soils threaten the safety and stability of foundations and structures, demanding advanced soil treatment techniques before construction can begin. We employ CFNN-based models to analyze and predict clay compressibility behavior, which is critical for assuring the efficacy and safety of construction operations on soft soils [27].

Furthermore, we believe that the proposal of deforce library will bring significant value and usefulness to other research groups, not only our group. Many recent publications use CFNN-based models, including [28–32]. This is also reflected in the download count on PyPI, where the deforce library has surpassed 1,000 downloads despite its recent released. In the future, we intend to enhance this library by including additional Derivative Free Optimization algorithms from various subgroups of DFO, rather than only global optimization (metaheuristics). Additionally, we aim to develop compatibility with different neural network types beyond CFNNs.

CRediT authorship contribution statement

Nguyen Van Thieu: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Hoang Nguyen:** Writing – review & editing, Writing – original

```
# Import libraries
from mealpy import StringVar, IntegerVar, MixedSetVar
from deforce import DfoTuneCfn

# Define boundary for hyperparameter of CFNN model
my_bounds = [
    IntegerVar(lb=5, ub=21, name="hidden_size"),
    StringVar(valid_sets=("relu", "leaky_relu", "celu", "prelu",
                          "gelu", "elu", "selu", "rrelu", "tanh",
                          "sigmoid"), name="act1_name"),
    StringVar(valid_sets=("relu", "leaky_relu", "celu", "prelu",
                          "gelu", "elu", "selu", "rrelu", "tanh",
                          "sigmoid"), name="act2_name"),
    IntegerVar(lb=700, ub=1000, name="max_epochs"),
    MixedSetVar(valid_sets=((8, 16, 32, 64)), name="batch_size"),
    StringVar(valid_sets=("Adadelta", "Adagrad", "Adam", "Adamax",
                          "AdamW", "ASGD", "LBFGS", "NAdam", "RAdam",
                          "RMSprop", "Rprop", "SGD"), name="optimizer"),
]

# Define Derivative free optimizes parameters
opt_paras = {"name": "WOA", "epoch": 10, "pop_size": 20}

# Define the Hybrid hyperparameter-optimized DFO-CFNN model
model = DfoTuneCfn(problem_type="regression", cv=3, scoring="MSE",
                   bounds=my_bounds,
                   optimizer="OriginalWOA",
                   optimizer_paras=opt_paras,
                   verbose=True, seed=42)
```

Listing 3: Snippet code to import and define the Hybrid hyperparameter-optimized DFO-CFNN models

draft, Validation, Project administration, Conceptualization. **Harish Garg:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources. **Gia Sirbiladze:** Writing – review & editing, Writing – original draft, Validation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by the Shota Rustaveli National Scientific Foundation of Georgia (SRNSF), grant number [STEM-22-226].

References

[1] B. Warsito, R. Santoso, Suparti, H. Yasin, Cascade forward neural network for time series prediction, J. Phys. Conf. Ser. 1025 (2018) 12097, http://dx.doi.org/10.1088/1742-6596/1025/1/012097.

[2] D. Narmandakh, C. Butscher, F. Doulati Ardejani, H. Yang, T. Nagel, R. Taherdangkoo, The use of feed-forward and cascade-forward neural networks to determine swelling potential of clayey soils, Comput. Geotech. 157 (2023) 105319, http://dx.doi.org/10.1016/j.compgeo.2023.105319.

[3] M. Alzayed, H. Chaoui, Y. Farajpour, Maximum power tracking for a wind energy conversion system using cascade-forward neural networks, IEEE Trans. Sustain. Energy 12 (2021) 2367–2377, http://dx.doi.org/10.1109/TSTE.2021.3094093.

[4] A.M.A. Shohda, M.A.M. Ali, G. Ren, J.-G. Kim, M.A.-E.-H. Mohamed, Application of cascade forward backpropagation neural networks for selecting mining methods, Sustainability 14 (2022) 635, http://dx.doi.org/10.3390/su14020635.

[5] S. Gündoğdu, T. Elbir, Application of feed forward and cascade forward neural network models for prediction of hourly ambient air temperature based on MERRA-2 reanalysis data in a coastal area of Turkey, Meteorol. Atmos. Phys. 133 (2021) 1481–1493, http://dx.doi.org/10.1007/s00703-021-00821-1.

[6] D.V. Wadkar, R.S. Karale, M.P. Wagh, Application of cascade feed forward neural network to predict coagulant dose, J. Appl. Water Eng. Res. 10 (2022) 87–100, http://dx.doi.org/10.1080/23249676.2021.1927210.

[7] X. Zheng, H. Nguyen, X.-N. Bui, Exploring the relation between production factors, ore grades, and life of mine for forecasting mining capital cost through a novel cascade forward neural network-based salp swarm optimization model, Resour. Policy. 74 (2021) 102300, http://dx.doi.org/10.1016/j.resourpol.2021.102300.

[8] H.H. Tan, K.H. Lim, Vanishing gradient mitigation with deep learning neural network optimization, in: 2019 7th Int. Conf. Smart Comput. Commun, IEEE, Sarawak, Malaysia, Malaysia, 2019, pp. 1–4, http://dx.doi.org/10.1109/ICSCC.2019.8843652.

[9] I. Mohd Yassin, R. Jailani, M.S.A. Megat Ali, R. Baharom, A.H. Abu Hassan, Z.I. Rizman, Comparison between cascade forward and multi-layer perceptron neural networks for NARX functional electrical stimulation (FES)-based muscle model, Int. J. Adv. Sci. Eng. Inf. Technol. 7 (2017) 215, http://dx.doi.org/10.18517/ijaseit.7.1.1388.

[10] T. Nguyen, T. Nguyen, B.M. Nguyen, G. Nguyen, Efficient time-series forecasting using neural network and opposition-based coral reefs optimization, Int. J. Comput. Intell. Syst. 12 (2019) 1144, http://dx.doi.org/10.2991/ijcis.d.190930.003.

[11] M. Yang, B. Xie, Y. Dou, G. Xue, Cascade forward artificial neural network based behavioral predicting approach for the integrated satellite-terrestrial networks, Mob. Networks Appl. 27 (2022) 1569–1577, http://dx.doi.org/10.1007/s11036-021-01875-6.

[12] L.M. Rios, N.V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global Optim. 56 (2013) 1247–1293, http://dx.doi.org/10.1007/s10898-012-9951-y.

[13] J.J. Moré, S.M. Wild, Benchmarking derivative-free optimization algorithms, SIAM J. Optim. 20 (2009) 172–191, http://dx.doi.org/10.1137/080724083.

[14] G. Di Pillo, G. Liuzzi, S. Lucidi, V. Piccialli, F. Rinaldi, A DIRECT-type approach for derivative-free constrained global optimization, Comput. Optim. Appl. 65 (2016) 361–397, http://dx.doi.org/10.1007/s10589-016-9876-3.

[15] C. Cartis, J. Fiala, B. Marteau, L. Roberts, Improving the flexibility and robustness of model-based derivative-free optimization solvers, ACM Trans. Math. Software 45 (2019) 1–41, http://dx.doi.org/10.1145/3338517.

[16] A.I.F. Vaz, L.N. Vicente, Pswarm: a hybrid solver for linearly constrained global derivative-free optimization, Optim. Methods Softw. 24 (2009) 669–685, http://dx.doi.org/10.1080/10556780902909948.

[17] N. Van Thieu, S. Mirjalili, MEALPY: An open-source library for latest meta-heuristic algorithms in Python, J. Syst. Archit. 139 (2023) 102871, http://dx.doi.org/10.1016/j.sysarc.2023.102871.

[18] B.M. Nguyen, B. Hoang, T. Nguyen, G. Nguyen, nQSV-net: a novel queuing search variant for global space search and workload modeling, J. Ambient Intell. Humaniz. Comput. 12 (2021) 27–46, http://dx.doi.org/10.1007/s12652-020-02849-4.

[19] A. Gaspar, D. Oliva, E. Cuevas, D. Zaldívar, M. Pérez, G. Pajares, Hyperparameter optimization in a convolutional neural network using metaheuristic algorithms, 2021, pp. 37–59, http://dx.doi.org/10.1007/978-3-030-70542-8_2.

[20] S. Al-Abri, T.X. Lin, M. Tao, F. Zhang, A derivative-free optimization method with application to functions with exploding and vanishing gradients, IEEE Control Syst. Lett. 5 (2021) 587–592, http://dx.doi.org/10.1109/LCSYS.2020.3004747.

[21] T. Nguyen, G. Nguyen, B.M. Nguyen, EO-CNN: An enhanced CNN model trained by equilibrium optimization for traffic transportation prediction, Procedia Comput. Sci. 176 (2020) 800–809, http://dx.doi.org/10.1016/j.procs.2020.09.075.

[22] S. Cahon, N. Melab, E.-G. Talbi, ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics, J. Heuristics 10 (2004) 357–380, http://dx.doi.org/10.1023/B:HEUR.0000026900.92269.ec.

[23] B.M. Nguyen, T. Tran, T. Nguyen, G. Nguyen, An improved sea lion optimization for workload elasticity prediction with neural networks, Int. J. Comput. Intell. Syst. 15 (2022) 90, http://dx.doi.org/10.1007/s44196-022-00156-8.

[24] N. Van Thieu, S.D. Barma, T. Van Lam, O. Kisi, A. Mahesha, Groundwater level modeling using augmented artificial ecosystem optimization, J. Hydrol. 617 (2023) 129034, http://dx.doi.org/10.1016/j.jhydrol.2022.129034.

[25] S. Mahdavi, M.E. Shiri, S. Rahnamayan, Metaheuristics in large-scale global continues optimization: A survey, Inf. Sci. (Ny). 295 (2015) 407–428, http://dx.doi.org/10.1016/j.ins.2014.10.042.

[26] N. Van Thieu, PerMetrics: A framework of performance metrics for machine learning models, J. Open Source Softw. 9 (2024) 6143, http://dx.doi.org/10.21105/joss.06143.

[27] Z. He, H. Nguyen, T.H. Vu, J. Zhou, P.G. Asteris, A. Mammou, Novel integrated approaches for predicting the compressibility of clay using cascade forward neural networks optimized by swarm- and evolution-based algorithms, Acta Geotech. 17 (2022) 1257–1272, http://dx.doi.org/10.1007/s11440-021-01358-8.

[28] G. Hayder, M.I. Solihin, H.M. Mustafa, Modelling of river flow using particle swarm optimized cascade-forward neural networks: A case study of kelantan river in Malaysia, Appl. Sci. 10 (2020) 8670, http://dx.doi.org/10.3390/app10238670.

[29] M. Hasanipanah, M. Jamei, A.S. Mohammed, M.N. Amar, O. Hocine, K.M. Khedher, Intelligent prediction of rock mass deformation modulus through three optimized cascaded forward neural network models, Earth Sci. Informat. 15 (2022) 1659–1669, http://dx.doi.org/10.1007/s12145-022-00823-6.

[30] F.A. Al Turki, M.M. Al Shammari, Predicting the output power of a photovoltaic module using an optimized offline cascade-forward neural network-based on genetic algorithm model, Technol. Econ. Smart Grids Sustain. Energy 6 (2021) 20, http://dx.doi.org/10.1007/s40866-021-00113-y.

[31] M.E. Abd-Elmaboud, H.A. Abdel-Gawad, K.S. El-Alfy, M.M. Ezzeldin, Estimation of groundwater recharge using simulation–optimization model and cascade forward ANN at east Nile Delta aquifer, Egypt, J. Hydrol. Reg. Stud. 34 (2021) 100784, http://dx.doi.org/10.1016/j.ejrh.2021.100784.

[32] C. Chiñas-Palacios, C. Vargas-Salgado, J. Aguila-Leon, E. Hurtado-Pérez, A cascade hybrid PSO feed-forward neural network model of a biomass gasification plant for covering the energy demand in an AC microgrid, Energy Convers. Manag. 232 (2021) 113896, http://dx.doi.org/10.1016/j.enconman.2021.113896.