Neurocomputing 517 (2023) 223–228

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original software publication

TSFE$_{DL}$: A python library for time series spatio-temporal feature extraction and prediction using deep learning

Ignacio Aguilera-Martos [a,c,*], Ángel M. García-Vico [a,c], Julián Luengo [a,c], Sergio Damas [b,c], Francisco J. Melero [b,c], José Javier Valle-Alonso [d], Francisco Herrera [a,c]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
[b] Department of Software Engineering, University of Granada, Granada, Spain
[c] Andalusian Institute of Data Science and Computational Intelligence (DaSCI), Spain
[d] Repsol Technology Lab, Spain

ARTICLE INFO

Article history:
Received 15 June 2022
Revised 13 September 2022
Accepted 24 October 2022
Available online 30 October 2022
Communicated by Zidong Wang

Keywords:
Time series
Deep learning
Python

ABSTRACT

The combination of convolutional and recurrent neural networks is a promising framework. This arrangement allows the extraction of high-quality spatio-temporal features together with their temporal dependencies. This fact is key for time series prediction problems such as forecasting, classification or anomaly detection, amongst others. In this paper, the TSFE$_{DL}$ library is introduced. It compiles 22 state-of-the-art methods for both time series feature extraction and prediction, employing convolutional and recurrent deep neural networks for its use in several data mining tasks. The library is built upon a set of `Tensorflow + Keras` and `PyTorch` modules under the `AGPLv3` license. The performance validation of the architectures included in this proposal confirms the usefulness of this Python package.

1. Introduction

The success of Machine Learning algorithms for time series prediction problems depends on the quality of the spatio-temporal features extracted. Deep Learning [1] models can produce non-linear transformations on data, yielding more abstract and useful spatio-temporal features and patterns than classical models for better prediction. Among the different areas of Deep Learning, the combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) is an important novelty for spatio-temporal feature extraction in time series [2–4]. Time series are a type of data sequentially sampled along a certain time interval. This structure implies the correlation of a sample with the previous time steps. The convolution operation enables the extraction of abstract high-level features obtaining refined local information. On the other hand, recurrent neural networks extract features that define the progress of a longer sequence of data. The combination of both is therefore fundamental to increase the performance and unify the extracted knowledge [5].

This neural network paradigm is providing interesting results in several areas, focusing the initial application on arrhythmia detection [6–8], with applications in other medical areas [9], energy forecasting [10,11] or remaining useful life prediction [12]. The absence of a unified collection of neural networks for time series generates the necessity of an easy-to-use and performant solution for the practitioner. The Python package TSFE $_{DL}$, presented in this paper, supports this process by providing a wide variety of easily customisable CNN-RNN Deep Learning models. All the available architectures are implemented from scratch, unifying the programming style and providing open source code for the included networks, presenting the user accessible and useful Python code.

The rest of the paper is structured as follows: Section 2 explains the software functionality and architecture. Section 3 gives instructions for the user to install the library and provides an example and an experimental framework for validating the library. Section 4 describes the quality standards of the code developing process. Finally, Section 5 summarises the conclusions of this paper and future work.

2. Software description

The TSFE $_{DL}$ library is built on Python 3. The library follows the programming style of the `Tensorflow + Keras` [13] functional API for further integration into state-of-the-art Machine Learning

Abbreviations: TS, Time series; DL, Deep Learning; Py, Python.

* Corresponding author at: Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain.
E-mail addresses: nacheteam@ugr.es (I. Aguilera-Martos), agvico@decsai.ugr.es (Á.M. García-Vico), julianlm@decsai.ugr.es (J. Luengo), sdamas@ugr.es (S. Damas), fjmelero@ugr.es (F.J. Melero), herrera@decsai.ugr.es (F. Herrera).

https://doi.org/10.1016/j.neucom.2022.10.062
0925-2312/

pipelines and seamless modification of the provided models. Moreover, a `PyTorch` [14] implementation of the library that relies on `PyTorch-Lightning` is provided to allow the user both to easily scale-up model execution in multi-GPU clusters and to create new architectures maximising code re-utilisation.

As can be seen in Fig. 1, the general architecture of the networks presented in this library is divided into two parts: the spatio-temporal embedding and the specialisation module. The former is in charge of extracting the most relevant spatio-temporal features and their bonds, whereas the specialisation module performs user-specific operations with the extracted features. This architecture allows the user to easily apply these networks to different data mining tasks just by the modification of the specialisation module. Similarly, the majority of spatio-temporal embeddings are composed of a set of CNN layers followed by a set of RNN layers such as LSTMs or GRUs. This structure enables the extraction of high-level spatio-temporal features together with their temporal dependencies. Additional details about the layers composition of each network are depicted in Table 1. The full explanation of the architectures can be found in Appendix A of the extended ArXiV version.[1]

The Keras models are implemented as Python functions. Each model has an assigned function which builds it and returns a Keras `Model` object. These functions contain several parameters to customise the architecture, returning either a model with no output layers configured or a classification group of layers with a given number of classes. After the model is returned from the function the practitioner can easily add more layers using the functional programming style of Keras.

The models implemented in Pytorch are structured as classes, inheriting from the template class `TSFEDL_BaseModule`. The template class implements the base methods to compose a PyTorch trainable network whereas each of the particular classes include the specific parameters to customise the module. These parameters include the `top_module` parameter, a PyTorch `nn.Module` object for the specialisation layers for the output of the model.

The goal of this library is to provide an easy source of models and code for the practitioners to use, gathering state-of-the art proposals for time series feature extraction. In order to achieve this, the models are programmed as reusable and customisable as possible, allowing the user to configure the output layers of the model to suit any type of problem such as classification, regression or forecasting. This feature is materialised by means of the `input_shape` and `top_module` parameters which control the input tensor shape and the output layers of the network.

In Table 1 the layers comprising each model are shown. A green tick represents the usage of that layer whereas a red cross indicates that it is not used. The five types of layers used are: one-dimensional convolution, long short-term memory recurrent layer, gated recurrent unit recurrent layer, bidirectional LSTM and bidirectional GRU.

3. Installation and quality standards

The TSFE $_{DL}$ library can be installed using PyPi using `pip install TSFEDL`. It is also available by cloning the repository from GitHub[2] and executing, from the root directory, the command `python setup.py install`. After that, the package will be available for its usage within the name `TSFEDL`.

The library code follows the `PEP8` style standard for Python. `Travis-CI` service is enabled in the repository of the project for continuous integration, ensuring back-compatibility and a proper operation of the architectures. *Semantic Versioning* and *Keep a Changelog* standards are integrated into the repository as well, making it easier for the users to notice the changes in the version progression. An extensive documentation is provided, following the `numpydoc` style of comments and using `sphinx` to generate it. The documentation is hosted in the `Read the Docs`[3] platform.

4. Cases of study

An experimental framework is proposed to study the performance of the networks and the capabilities of the library. In this section, a new model leveraging the characteristics of TSFE $_{DL}$ is created, remarking the customisation capability for any given task. The objective of the custom model is to extract new spatio-temporal features to outperform the included architectures in the library on several kinds of problems. The problems analysed are briefly described below (the extended explanation can be found in Appendix B of the extended ArXiV version [4]): (See Fig. 2).

1. Forecasting. The aim is to predict the following $n = 50$ time-steps of a time series from the Spanish Digital Seismic Network (IGN) [31] to identify future earthquakes.[5]
2. Classification. The objective is to classify segments of ECG signals from the MIT-BIH dataset [32] to identify different types of cardiac arrhythmia.[6]
3. Anomaly Detection. The goal is to identify malicious attacks from network traffic using the KDD Cup '99 dataset [33].[7]

The network shown in Fig. 3 is a modification of the `HuangMeiLing` model which includes a new LSTM layer. Next, the spatio-temporal features extracted by this model will be further processed using a specialisation module to fulfill the requirements of each task. In Fig. 3, an example for the forecasting task is shown. Additional details about these experiments and their characteristics can be found in the library repository.[8] We must highlight that this specialisation module is employed on all the networks of the library as shown in Fig. 3 to perform a fair comparison between methods.

The results obtained from each method on each task are shown in Table A.1. These results led us to the conclusion that the models in this library can be applied to different tasks successfully, even if they have not been initially designed for them. In fact, no method outperforms all the rest for all of the problems.

From the results we can see that there are some common characteristics among the best performing networks for each problem. For the prediction problem the three best networks are HuangMeiLing, ShiHaotian and ZhengZhenyu. These networks are characterized by having a high number of convolutions but not having more than one recurrent layer or even none at all. For the classification problem the best performing networks are HtetMyetLynn, WeiXiaoyan and YiboGao. These networks include attention mechanisms, bidirectional recurrent layers or simply more than one recurrent layer. No common pattern is found for all of them but we can observe that long-term temporal dependencies are more important than in the prediction problem. Finally, for the anomaly detection problem, we can see that the best performing networks are YiboGao, OhShuLih and ChenChen. From these results it can be seen that this problem requires a treatment halfway between the classification problem and the prediction problem. These networks have either attention mechanisms or a large number of con-

[1] Aguilera-Martos et al. (https://arxiv.org/abs/2206.03179 Appendix A).
[2] https://github.com/ari-dasci/S-TSFE-DL.
[3] https://s-tsfe-dl.readthedocs.io/en/latest/.
[4] Aguilera-Martos et al. (https://arxiv.org/abs/2206.03179 Appendix B)
[5] IGN Data under contact https://www.ign.es/web/ign/portal/sis-area-sismicidad
[6] MIT-BIH data https://physionet.org/content/mitdb/1.0.0/
[7] KDDCup99 data http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[8] https://github.com/ari-dasci/S-TSFE-DL/tree/main/examples.

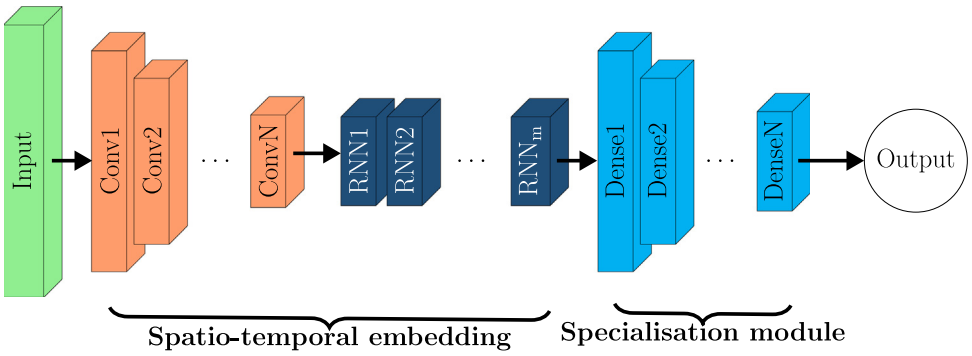

Fig. 1. General scheme of the models presented in the TSFE $_{DL}$ library.

Table 1
TSFE$_{DL}$ models and architecture type.

Model names	CNN	LSTM	GRU	Bid. LSTM	Bid. GRU
CaiWenjuan [15]	✓	X	X	X	X
ChenChen [8]	✓	✓	X	X	X
FuJiangmeng [16]	✓	✓	X	X	X
GaoJunli [7]	X	✓	X	X	X
GenMinxing [17]	X	X	X	✓	X
HongTan [18]	✓	✓	X	X	X
HtetMyetLynn [19]	✓	X	X	✓	✓
HuangMeiLing [20]	✓	X	X	X	X
KhanZulfiqar [10]	✓	X	✓	X	X
KimTaeYoung [11]	✓	✓	X	X	X
KongZhengmin [12]	✓	✓	X	X	X
LihOhShu [21]	✓	✓	X	X	X
OhShuLih [6]	✓	✓	X	X	X
ShiHaotian [22]	✓	✓	X	X	X
WangKejun [23]	✓	✓	X	X	X
WeiXiaoyan [9]	✓	✓	X	X	X
YaoQihang [24]	✓	✓	X	X	X
YiboGao [25]	✓	X	X	X	X
YildirimOzal [26]	✓	✓	X	X	X
ZhangJin [27]	✓	X	X	X	✓
ZhengZhenyu [28]	✓	✓	X	X	X
SharPar [29]	✓	✓	X	X	X
DaiXiLi [30]	✓	X	X	X	X

volutional and recurrent layers, to model local features and long-term dependencies.

The custom model created in Fig. 3 outperforms the rest of the networks in the forecasting task, confirming that the customisation capability can enhance the performance of the networks. This means that the library's variety of models, together with its customisation, gives great flexibility concerning the target application.

5. Concluding remarks

The Python library TSFE $_{DL}$ gathers 22 Deep Learning state-of-the-art methods combining both convolutional and recurrent layers. The implementation relies on the `Keras` functional API and `PyTorch-Lightning` to easily integrate the algorithms into state-of-the-art Machine Learning pipelines. This fact, together with the architecture of the library, allows us to easily create and customise Deep Learning models for different data mining tasks.

The library provides performant, expandable and customisable neural networks being proved the usefulness of the provided piece of software. The results confirm that the included models can be successfully applied to different tasks. Therefore, the application scope of this type of models can be significantly extended. This Python module stands as a convenient solution for the practitioner.

As future work, this library is under constant development, including recently published architectures of relevance. In addition, it is planned to expand the functionality of this library by adding pre-trained models as well as visualisation modules.

CRediT authorship contribution statement

Ignacio Aguilera-Martos: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing. **Ángel M. García-Vico:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing-review-editing. **Julián Luengo:** Conceptualization, Resources, Supervision, Writing – review & editing. **Sergio Damas:** Conceptualization, Formal analysis, Resources, Writing – review & editing. **Francisco J. Melero:** Supervision, Investigation, Conceptualization, Resources, Writing – review & editing. **José**

```
# Installation via PyPi
pip install tsfedl
# Installation via GitHub repository
curl -L -O https://github.com/ari-dasci/S-TSFE-DL/archive/refs/heads/main.zip
unzip main.zip
cd S-TSFE-DL-main
python setup.py install
```

Fig. 2. Installation procedure of the library TSFE $_{DL}$.

```python
import tensorflow as tf
import TSFEDL.models_keras as TSFEDL
# Create the new model
input = tf.keras.Input(shape=(1000, 1))
model = TSFEDL.HuangMeiLing(input_tensor=input, include_top=False)
x = model.output
x = tf.keras.layers.LSTM(units=20)(x)
# Add the specilisation module
# this is for forcasting only
# it is unique for each task
x = tf.keras.layers.Flatten(x)
x = tf.keras.layers.Dense(50)(x)
out = tf.keras.layers.Reshape([50,1])(x)
# Create the Keras model and train it.
new_model = tf.keras.Model(inputs=input, outputs=out)
new_model.compile(loss='mae', optimizer='adam', metrics=['mae'])
new_model.fit(data, epochs=20)
```

Fig. 3. Creation of a new CNN-RNN model for time series forecasting using TSFE $_{DL}$.

Javier Valle-Alonso: Data curation, Methodology. **Francisco Herrera:** Funding acquisition, Project administration, Supervision, Resources, Writing – review & editing.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jose Javier Valle Alonso reports a relationship with Repsol SA that includes: employment.

Acknowledgements

This work has been partially supported by the Contract UGR-AM OTRI-4260 and the Regional Government of Andalusia, under the program "Personal Investigador Doctor", reference DOC_00235. This work was also supported by project PID2020-119478 GB-I00 granted by Ministerio de Ciencia, Innovación y Universidades, and projects P18-FR-4961 and P18-FR-4262 by Proyectos I + D+i Junta de Andalucia 2018.

Appendix A. Cases of study experimental results

See Table A.1.

Table A.1
Results extracted from the TSFE $_{DL}$ models for the three tasks analysed.

Model	Classification (Accuracy) MIT-BIH arrythmia	Time series forecasting (MAE) IGN	Anomaly detection (AUC) KDD Cup99
CaiWenjuan [15]	0.6845	3.8784	0.5945
ChenChen [8]	0.9233	151.3446	**0.7402**
DaiXiLi [30]	0.9563	151.3807	0.6330
FuJiangmeng [16]	0.5612	12.0492	0.5135
GaoJunLi [7]	0.4996	45.9238	0.4821
GenMinxing [17]	0.9569	6.8440	0.5186
HongTan [18]	0.8412	9.1740	0.6069
HtetMyetLynn [19]	0.9709	12.1476	0.4866
HuangMeiLing [20]	0.9633	1.4135	0.5180
KhanZulfiqar [10]	0.9400	29.9031	0.6272
KimTaeYoung [11]	0.6378	3.1309	0.5586
KongZhengmin [12]	0.6515	2.0443	0.5535
LihOhShu [21]	0.8196	3.0234	0.5302
OhShuLih [6]	0.7224	2.2590	0.7103
SharPar [29]	0.9545	118.1454	0.5304
ShiHaotian [22]	0.9581	1.9449	0.6071
WangKejun [23]	0.9539	3.0436	0.4897
WeiXiaoyan [9]	0.9703	2.9720	0.6462
YaoQihang [24]	0.9681	2.7898	0.5939
YiboGao [25]	**0.9718**	149.2162	0.7356
YildirimOzal [26]	0.9075	44.5272	0.6988
ZhangJin [27]	0.9575	30.6660	0.5605
ZhengZhenyu [28]	0.9196	2.0847	0.5374
Model of the example (Fig. 3)	0.9248	**1.1402**	0.5089

Appendix B. Required metadata

.

Appendix C. Current code version

See Table A.2.

Table A.2
Code metadata (mandatory).

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v1.0.3
C2	Permanent link to code/ repository used of this code version	https://github.com/ari-dasci/S-TSFE-DL
C3	Legal Code License	GNU Affero General Public License v3.0
C4	Code versioning system used	Git
C5	Software code languages, tools, and services used	Python 3, Keras, Tensorflow, PyTorch and PyTorch-Lightning
C6	Compilation requirements, operating environments & dependencies	OS-X, Unix-like or Microsoft Windows, a Python interpreter (3.7) and the following Python packages: pytorch-lightning, scikit-learn, tensorflow-gpu/tensorflow, torchmetrics, wfdb, obspy
C7	If available Link to developer documentation/manual	https://s-tsfe-dl.readthedocs.io/en/latest/
C8	Support email for questions	nacheteam@ugr.es

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[2] J.H. Tan, Y. Hagiwara, W. Pang, I. Lim, S.L. Oh, M. Adam, R.S. Tan, M. Chen, U.R. Acharya, Application of stacked convolutional and long short-term memory network for accurate identification of cad ecg signals, Comput. Biol. Med. 94 (2018) 19–26.
[3] H. Lu, Z. Ge, Y. Song, D. Jiang, T. Zhou, J. Qin, A temporal-aware lstm enhanced by loss-switch mechanism for traffic flow forecasting, Neurocomputing 427 (2021) 169–178.
[4] M.O. Alassafi, M. Jarrah, R. Alotaibi, Time series predicting of covid-19 based on deep learning, Neurocomputing 468 (2022) 335–344.
[5] P. Zhang, Y. Hang, X. Ye, P. Guan, J. Jiang, J. Tan, W. Hu, A united cnn-lstm algorithm combining rr wave signals to detect arrhythmia in the 5g-enabled medical internet of things, IEEE Internet Things J. 9 (16) (2022) 14563–14571.
[6] S.L. Oh, E.Y.K. Ng, R. San Tan, U.R. Acharya, Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats, Comput. Biol. Med. 102 (2018) 278–287.
[7] J. Gao, H. Zhang, P. Lu, Z. Wang, An effective LSTM recurrent network to detect arrhythmia on imbalanced ECG dataset, J. Healthcare Eng. 2019 (2019) 6320651.
[8] C. Chen, Z. Hua, R. Zhang, G. Liu, W. Wen, Automated arrhythmia classification based on a combination network of CNN and LSTM, Biomed. Signal Process. Control 57 (2020).
[9] X. Wei, L. Zhou, Z. Zhang, Z. Chen, Y. Zhou, Early prediction of epileptic seizures using a long-term recurrent convolutional network, J. Neurosci. Methods 327 (2019).
[10] M. Sajjad, Z.A. Khan, A. Ullah, T. Hussain, W. Ullah, M.Y. Lee, S.W. Baik, A novel CNN-GRU-based hybrid approach for short-term residential load forecasting, IEEE Access 8 (2020) 143759–143768.
[11] T.-Y. Kim, S.-B. Cho, Predicting residential energy consumption using CNN-LSTM neural networks, Energy 182 (2019) 72–81.
[12] Z. Kong, Y. Cui, Z. Xia, H. Lv, Convolution and long short-term memory hybrid deep neural networks for remaining useful life prognostics, Appl. Sci. 9 (19) (2019) 4156.
[13] F. Chollet, Others, Keras, https://keras.io (2015).
[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems, 32, 2019, pp. 8024–8035.
[15] W. Cai, Y. Chen, J. Guo, B. Han, Y. Shi, L. Ji, J. Wang, G. Zhang, J. Luo, Accurate detection of atrial fibrillation from 12-lead ECG using deep neural network, Comput. Biol. Med. 116 (2020).
[16] J. Fu, C. Sun, Z. Yu, L. Liu, A hybrid CNN-LSTM model based actuator fault diagnosis for six-rotor UAVs, 2019 Chinese Control And Decision Conference (CCDC), IEEE (2019) 410–414.
[17] M. Geng, W. Zhou, G. Liu, C. Li, Y. Zhang, Epileptic seizure detection based on stockwell transform and bidirectional long short-term memory, IEEE Trans. Neural Syst. Rehabil. Eng. 28 (3) (2020) 573–580.
[18] J.H. Tan, Y. Hagiwara, W. Pang, I. Lim, S.L. Oh, M. Adam, R.S. Tan, M. Chen, U.R. Acharya, Application of stacked convolutional and long short-term memory network for accurate identification of cad ecg signals, Comput. Biol. Med. 94 (2018) 19–26.
[19] H.M. Lynn, S.B. Pan, P. Kim, A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks, IEEE Access 7 (2019) 145395–145405.
[20] M.-L. Huang, Y.-S. Wu, Classification of atrial fibrillation and normal sinus rhythm based on convolutional neural network, Biomed. Eng. Lett. 10.
[21] O.S. Lih, V. Jahmunah, T.R. San, E.J. Ciaccio, T. Yamakawa, M. Tanabe, M. Kobayashi, O. Faust, U.R. Acharya, Comprehensive electrocardiographic diagnosis based on deep learning, Artif. Intell. Med. 103 (2020).
[22] H. Shi, C. Qin, D. Xiao, L. Zhao, C. Liu, Automated heartbeat classification based on deep neural network with multiple input layers, Knowl.-Based Syst. 188 (2020).
[23] K. Wang, X. Qi, H. Liu, Photovoltaic power forecasting based LSTM-Convolutional Network, Energy 189 (2019).
[24] Q. Yao, R. Wang, X. Fan, J. Liu, Y. Li, Multi-class arrhythmia detection from 12-lead varied-length ECG using attention-based time-incremental convolutional neural network, Inf. Fusion 53 (2020) 174–182.
[25] Y. Gao, H. Wang, Z. Liu, An end-to-end atrial fibrillation detection by a novel residual-based temporal attention convolutional neural network with exponential nonlinearity loss, Knowl.-Based Syst. 212 (2021).
[26] O. Yildirim, U.B. Baloglu, R.-S. Tan, E.J. Ciaccio, U.R. Acharya, A new approach for arrhythmia classification using deep coded features and LSTM networks, Comput. Methods Programs Biomed. 176 (2019) 121–133.
[27] J. Zhang, A. Liu, M. Gao, X. Chen, X. Zhang, X. Chen, ECG-based multi-class arrhythmia detection using spatio-temporal attention-based convolutional recurrent neural network, Artif. Intell. Med. 106 (2020).
[28] Z. Zheng, Z. Chen, F. Hu, J. Zhu, Q. Tang, Y. Liang, An automatic diagnosis of arrhythmias using a combination of CNN and LSTM technology, Electronics 9 (1) (2020) 121.
[29] G. Sharma, A. Parashar, A.M. Joshi, Dephnn: A novel hybrid neural network for electroencephalogram (eeg)-based screening of depression, Biomed. Signal Process. Control 66 (2021).
[30] J. Dai, X. Xi, G. Li, T. Wang, Eeg-based emotion classification using improved cross-connected convolutional neural network, Brain Sci. 12(8).
[31] International Federation of Digital Seismograph Networks, Spanish Digital Seismic Network. Dataset/Seismic Network. doi:10.7914/SN/ES.
[32] G.B. Moody, R.G. Mark, The impact of the MIT-BIH arrhythmia database, IEEE Eng. Med. Biol. Mag. 20 (3) (2001) 45–50.
[33] S. Hettich, S.D. Bay, The uci kdd archive, Irvine, CA: University of California, Department of Information and Computer Science. URL: http://kdd.ics.uci.edu.

Ignacio Aguilera is currently working in his PhD in Computer Science in the University of Granada, which he started in 2020. He received his B.Sc. in Mathematics and his B.Sc. in Computer science in 2019 and his M.Sc. in Data Science and Computer Engineering in 2020, all from the University of Granada, Granada, Spain. His research interests include anomaly detection, Deep Learning and Machine Learning.

Ángel Miguel García-Vico B.Sc. degree in Computer Science from University of Jaén, Spain, in 2015 with Extraordinary Award. M.Sc. Data Science and Computer Engineering from the University of Granada, Spain, in 2016. PhD. in Computer Science from the University of Jaén in 2020. He is currently an Assistant Professor with the Control and Communication Systems Dept., UNED, Spain. He is the author of 10 research papers and more than 10 contributions to international conferences. His research interests include supervised descriptive rule discovery, deep learning, big data and data stream mining.

Julián Luengo received the M.S. degree in computer science and the Ph.D. from the University of Granada, Granada, Spain, in 2006 and 2011 respectively. He currently acts as an Assistant Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada, Spain. His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, noisy data, data complexity and fuzzy systems. Dr. Luengo has been given some awards and honors for his personal work or for his publications in and conferences, such as IFSA-EUSFLAT 2009 Best Student Paper Award. He belongs to the list of the Highly Cited Researchers in the area of Computer Sciences (2015–2018): http://highlycited.com/ (Clarivate Analytics).

Sergio Damas received his Ph.D. in Computer Science in 2003 from the University of Granada, where he has been a Professor at the Software Engineering Department since 2018. He has co-authored 150+ scientific publications including 40+ articles indexed in the JCR. He has been principal investigator of numerous private research contracts and research projects in competitive regional, national and international calls. He has received different national and international awards. His current research interests are related to the application of artificial intelligence to complex problems of very different fields such as forensic anthropology, medical imaging and marketing.

Francisco Javier Melero is M.Sc. in Computer Engineering (2001), M.Sc. in Data Science (2020) and PhD in Computer Graphics (2008) by the University of Granada. He is currently Assistant Professor at the Software Engineering Department of the University of Granada, and researcher of DaSCI, the Andalusian Research Institute for Data Science and Computational Intelligence. His research interests include mainly computer graphics, data mining and knowledge discovery, anomaly detection and digital twins. He has a wide experience in applying information technologies to cultural heritage and tourism, through applied research and as founder of startups. He has served at the Executive Board of Eurographics as assistant treasurer (2005-2014) and chaired several international conferences (CAA2010, EG-VCBM2018, EG-GCH2020).

Jose Javier Valle-Alonso is currently working in Repsol Technology Lab as Scientist in Spain. He received his B. Sc. in Physics in 2017 from the U.N.E.D. He worked as lab technician in Holmen Paper and Optiroc in 2002 and 2003. Since May 2003 is working in Repsol. His research interests include artificial intelligence and applied physics.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and Director of the Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI). He's an academician in the Royal Academy of Engineering (Spain).
He has been the supervisor of 54 Ph.D. students. He has published more than 600 journal papers, receiving more than 118000 citations (Scholar Google, H-index 164). He has been nominated as a Highly Cited Researcher (in the fields of Computer Science and Engineering, respectively, 2014 to present, Clarivate Analytics). He acts as editorial member of a dozen of journals.
His current research interests include among others, computational intelligence, information fusion and decision making, explainable artificial intelligence and data science (including data preprocessing, prediction and big data).