EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

Research in Computing Science

Vol. 151 No. 3
March 2022

Research in Computing Science

Research in Computing Science, Año 21, Volumen 151, No. 3, marzo de 2022, es una publicación mensual, editada por el Instituto Politécnico Nacional, a través del Centro de Investigación en Computación. Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othon de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, Ciudad de México, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor responsable: Dr. Grigori Sidorov. Reserva de Derechos al Uso Exclusivo del Título No. 04-2019-082310242100-203. ISSN: en trámite, ambos otorgados por el Instituto Politécnico Nacional de Derecho de Autor. Responsable de la última actualización de este número: el Centro de Investigación en Computación, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othon de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Fecha de última modificación 01 de marzo de 2022.

Research in Computing Science, year 21, Volume 151, No. 3, March 2022, is published monthly by the Center for Computing Research of IPN.

Hybrid Intelligent Systems

Martín Montes Rivera
Carlos Alberto Ochoa Ortiz
Julio César Ponce Gallegos
Edgar Gonzalo Cossio Franco (eds.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2022

ISSN: in process

Copyright © Instituto Politécnico Nacional 2022
Formerly ISSNs: 1870-4069, 1665-9899.

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX, DBLP and Periodica

Electronic edition

ISSN 1870-4069

Table of Contents

ISSN 1870-4069

Deep Learning based Failure Prognostic Model for Aerospace Propulsion Systems

Jose Nava, Alberto Ochoa

Universidad Autónoma de Ciudad Juárez,
Mexico

jose.nava@3pillarglobal.com, alberto.ochoa@uacj.mx

Abstract. The continuous growth of complexity in aerospace systems is making it increasingly difficult to analyze and process information from sensors in propulsion systems in real-time to obtain a good model for predicting failures in aviation turbofan units. In this work, a model of Neural Networks with deep learning to predict the Remaining Useful Life (RUL) in aerospace propulsion systems is proposed. Specifically, a Turbofan (jet or turbojet engine) and its set of sensors are analyzed to represent and mathematically predict the evolution of the failure state and engine degradation. A data set from NASA (National Aeronautics and Space Administration) has been used, found in the NASA Prognostics Center of Excellence (PCoE) repository at the Ames Research Center. This data set has been generated with the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) software, which is a dynamic model created by this institution. In addition, the TensorFlow platform has been used to program preprocessing, analysis, and create a Deep Learning model. The importance of this model goes beyond its usefulness in the aerospace industry, since the main functions of a gas turbine such as air compression, combustion, and energy recovery, allow these systems to be used for multiple purposes; therefore, this model can be replicated in gas turbines in other industries such as thermoelectric plants, petrochemical plants, in propulsion in the military and maritime industries, and in cargo ships.

Keywords: Remaining Useful Life (RUL), Long Short-Term Memory (LSTM), flight envelope.

1 Introduction

1.1 Background

In Artificial Intelligence, machine learning and deep learning can automatically learn hierarchical representations of data on a large scale, which makes these, effective tools for failure prediction applications within predictive maintenance, especially in the presence of industrial multidimensional and high-volume data. Traditional data-driven mathematical strategies require manual feature extraction and specific feature selection processes, which is highly dependent on programmers' experience and knowledge of signal processing in electronics [12]. Furthermore, aviation systems currently send information directly from fuselage sensors to Electronic Flight Instruments (EFI), and these instruments only evaluate the information from the sensors within a predetermined range, that is, they do not analyze correlation information or patterns between the different dimensions to implement multivariate analysis in real time and predict effectively.

Conventional frameworks cannot be updated in real time and require a lot of work when dealing with large-scale data sets. In comparison, a deep learning algorithm makes it possible to integrate tasks such as feature extraction, feature selection, and regression into a dynamic, hybrid architecture, making automation of such complex predictions ideal for these systems. There are ongoing Aerospace Hardware product development efforts in the industry where it is possible to implement the model proposed in this work.

1.2 Aerospace Propulsion Systems

Propulsion by gas exhaust (or jet propulsion) can be defined as the force that is generated in the opposite direction to the expulsion of the gases. In a turbine engine, the intake, compression, combustion, and exhaust functions take place in the same combustion chamber.

Consequently, each of these functions must have an exclusive occupation of the chamber during its respective part of the combustion cycle [9]. An important feature of the gas turbine engine is that, in its design, separate sections are dedicated to each function, and all functions are performed simultaneously without interruption, hence the need for intensive sensors monitoring [4].

2 NASA C-MAPSS Dataset Description

As mentioned previously in this work, a dataset created by NASA has been used, generated with the dynamic model of Simulation of Commercial Modular Aeropropulsion Systems (C-MAPSS). The damage propagation modeling used to generate this synthetic dataset builds on the modeling strategy from previous work and incorporates two new levels of fidelity. First, considering the actual flight conditions recorded on board of a commercial aircraft. Secondly, it extends the degradation model by relating the degradation process to operation history [1].

The CMAPSS dynamic model is a high-fidelity computer model for the simulation of a large and realistic commercial turbofan engine. A schematic representation of the engine is shown in Figure 1, where many of the turbine sensors to be processed can be observed. In addition to the thermodynamic model of the engine, the package includes an atmospheric model capable of operating at altitudes from sea level to 40,000 feet, Mach numbers from 0 to 0.90 and sea level temperatures from -60 to 103 degrees F. The CMAPSS software also includes a power management system that allows the engine to run in a wide range of thrust levels across the full range of flight conditions.

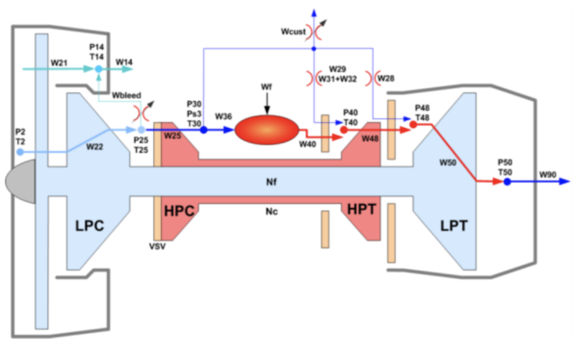


Fig. 1. Schematic representation of C-MAPSS Model from NASA.

Table 1. Descriptor Variables of Flight Envelope.

Index	Symbol	Description	Units
1	alt	Altitude	ft
2	Mach	Flight Mach number	-
3	TRA	Throttle-resolver angle	%

The NASA dataset provides synthetic run-to-failure degradation trajectories from optimal turbine condition until failure for a small fleet, comprising nine turbofan engines with unknown and different initial health conditions.

The actual flight conditions as they would be recorded on board a commercial aircraft (also known as the operating mode) were taken as input for the C-MAPSS model. Table 1 presents the scenario descriptor variables that describe the flight conditions, which is called Flight Envelope.

Within the NASA dataset, according to their structural and simulation analysis, contemplates that the datasets consist of multiple multivariate time series. Each dataset is in turn divided into training and testing subsets. Each time series comes from a different engine, that is, the data can be considered to come from a fleet of engines of the same type. Each engine starts with different degrees of initial degradation and manufacturing variations unknown to the user. These degradations and variations are considered normal, that is, it is not considered a fault condition. There are three operating settings/modes that have a substantial effect on engine performance. These adjustments are also included in the data. The data is contaminated with sensor noise.

The engine runs normally at the beginning of each time series and begins to degrade sometime during the series. In the training set, the degradation grows in magnitude until a predefined threshold is reached beyond which it is not preferable to operate the motor. In the test set, the time series ends some time before complete degradation. The objective is to predict the number of remaining operating cycles earlier in the test set, that is, the number of operating cycles after the last cycle that the engine will continue to operate normally.

Table 2. Dataset variables description.

No	Variable
1	Turbofan unit number
2	Time, in cycles
3	Operational setting 1
4	Operational setting 2
5	Operational setting 3
6	Sensor measurement 1
7	Sensor measurement 2
8	Sensor measurement 3
…	…
26	Sensor measurement 21

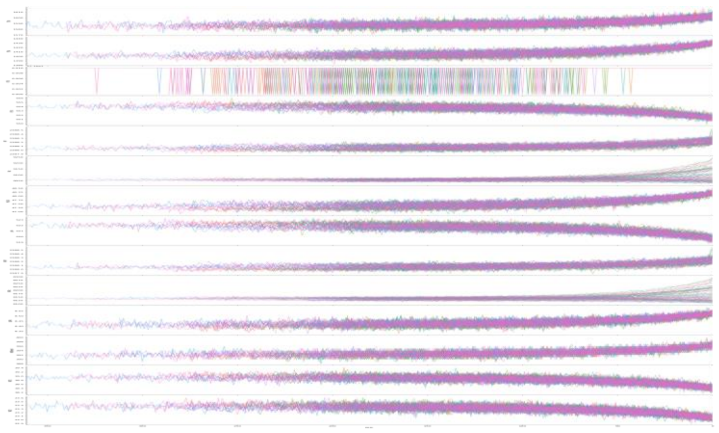


Fig. 2. Charts for all sensors of all turbofan units from cycle 1, until failure.

The data is provided by NASA as a compressed zip text file with 26 columns of numbers, separated by spaces.

Table 2 represents the structure of the columns in the dataset.

The NASA datasets used were PHM08 and the Turbofan Engine Degradation Simulation Data Set.

The main characteristics of PHM08 dataset are presented below:

– Consists of 45,918 records
– Training dataset trajectories: 218 different turbofans where each one fails at a different point.

– Test dataset trajectories: 218 different turbofans where each one fails at a different point.

As explained in the NASA repository instructions, the three operational settings stated previously in this work are also included in the data.
To work with this data, the following Python libraries were used:

– TensorFlow, used for numerical calculation, for deep learning models.
– SKLearn, used for machine learning.
– Pandas, used for numerical data analysis, whose function is to create a data frame structure to be able to work with it.
– Numpy, used for linear algebra.
– Matplotlib, used for graphics.
– mpl_toolkits, used for graphics.

Sensor's selection process is critical to improve aircraft engine diagnostics [10]. During the first analysis of the dataset, it has been observed that the last two columns are null data, so those columns have been removed from the data frame as a first strategy.

After this, an array was created with the names of the columns of the dataset, to be able to add them to the structure of the data frame, since the dataset comes as raw data, without structure.
Next, descriptive statistics was applied in an exploratory analysis, and the most relevant sensors were plotted against the RUL variable (the number of cycles remaining for the turbofan to reach a failure state), where it can be observed that it starts at 357, which is the greater number of cycles of a turbofan in the dataset, up to where RUL is 0. Something interesting to observe is the density of the graph at different times in cycles.

It can be observed that the X axis shows the RUL variable from the highest, which is the maximum number of cycles (357), where there is less density in the graph due to the existence of fewer sensors with many cycles, until failure (RUL = 0). After looking at the behavior of all 21 sensors, 6 sensors behave with constant values during all cycles, so the following sensors ["T2", "P2", "epr", "farB", "Nf_dmd", "PCNfR_dmd"] were then extracted from the dataset.
StandardScaler class from SkLearn Machine Learning library has been used to standardize the characteristics by eliminating the mean and scaling to the variance unity.
The standard result of a sample x is calculated as follows:

$$z = (x - u) / s, \tag{1}$$

where ***u*** is the mean of the training samples or zero if ***with_mean = False***, and ***s*** is the standard deviation of the training samples or one if ***with_std = False***.

3 Experiments and Creation of the Neural Network Model

In this work two hypotheses have been proposed that will be later supported with experiments: (H1) An LSTM architecture is more powerful than a traditional multilayer neural network when applied to multivariate time series prediction tasks, and when is included in an automated prediction framework such as the one in the present work, it can outperform multivariate time series predictions on RUL prediction reference data sets. The alternate hypothesis (H2) states that a traditional multilayer neural network outperforms an LSTM architecture in multivariate time series predictions on RUL prediction reference data sets.

For this type of problem, a deep recurrent neural network (RNN) is used to learn the multivariate time series regression function. It is important to state three complex conditions of the problem, which are the multiple operating conditions, the different operating behavior between sensors in terms of the range of parameter values, and the lack of knowledge of the exact starting point of failure or degradation.

In recent years, deep recurrent neural networks (RNN) based on gated units such as Long Short Term Memory [3] have been used successfully to model sequential data. RNNs have been shown to model the temporal (sequential) aspect of sensor data, as well as capture inter-sensor dependencies. The RNNs have been used to model the behavior of motors as a function of time series of multiple sensors with applications for the detection of anomalies and faults [5].

An LSTM unit maintains a cell state using an entry gate, a forget gate, and an exit gate: at a given time step, the entry gate decides what should be added to the cell state, the forget gate decides what should be removed from the state cell, and the output gate decides what part of the cell state should be the output from the LSTM unit [2].

In the equations below, column vectors are denoted with lowercase in bold and matrices with uppercase in bold. For a hidden layer with ***h*** LSTM units, the values for the input gate $\boldsymbol{i_t}$, the forget gate $\boldsymbol{f_t}$, the output gate $\boldsymbol{o_t}$, the hidden state $\boldsymbol{z_t}$, and the state of cell $\boldsymbol{c_t}$ at time ***t*** are calculated using the input current $\boldsymbol{x_t}$, previous hidden state $\boldsymbol{z_{t-1}}$, and cell state $\boldsymbol{c_{t-1}}$, where $\boldsymbol{i_t}$, $\boldsymbol{f_t}$, $\boldsymbol{o_t}$, $\boldsymbol{z_t}$, and $\boldsymbol{c_t}$ are h-dimensional vectors with real values so that:

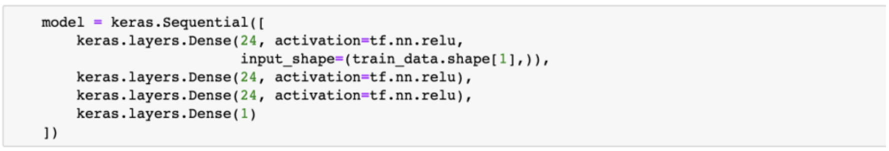
$$\boldsymbol{zt = f\ (xt,\ zt\text{-}1,\ ct\text{-}1),} \tag{1}$$

as indicated in the following equation [11]:

$$\begin{pmatrix} i_t^l \\ f_t^l \\ o_t^l \\ g_t^l \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W_{2h,4h} \begin{pmatrix} D(z_t^{l-1}) \\ z_t^{l-1} \end{pmatrix}, \tag{1}$$

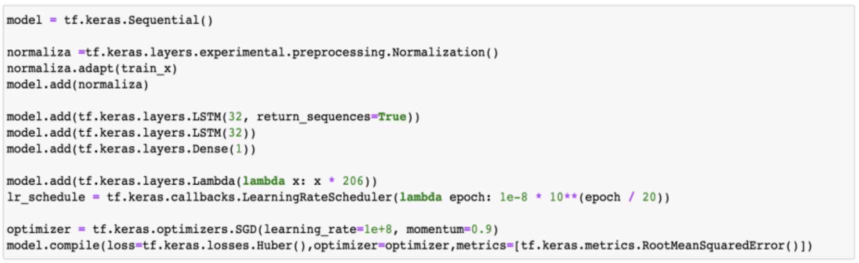
where the time series goes through the following transformations iteratively in the *l-th* hidden layer for ***t = 1*** to ***T***, where ***T*** is the length of the time series.

LSTM has the powerful ability to remove or add information to the state of the cell, carefully regulated by these structures called gates [7]. Gates are a way to optionally pass information. They are composed of a sigmoid neural network layer and a point multiplication operation. The sigmoid layer generates numbers between zero and one, which describes how much of each component should be allowed to pass. A value of zero means "let nothing go", while a value of one means "let everything go." An LSTM has three of these doors to protect and control the state of the cell [8].



```
model = keras.Sequential([
    keras.layers.Dense(24, activation=tf.nn.relu,
                       input_shape=(train_data.shape[1],)),
    keras.layers.Dense(24, activation=tf.nn.relu),
    keras.layers.Dense(24, activation=tf.nn.relu),
    keras.layers.Dense(1)
])
```

Fig. 3. First Neural model with a traditional Sequential Layer Dense layer.



```
model = tf.keras.Sequential()

normaliza =tf.keras.layers.experimental.preprocessing.Normalization()
normaliza.adapt(train_x)
model.add(normaliza)

model.add(tf.keras.layers.LSTM(32, return_sequences=True))
model.add(tf.keras.layers.LSTM(32))
model.add(tf.keras.layers.Dense(1))

model.add(tf.keras.layers.Lambda(lambda x: x * 206))
lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(learning_rate=1e+8, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),optimizer=optimizer,metrics=[tf.keras.metrics.RootMeanSquaredError()])
```

Fig. 4. Second Neural model with 2 LSTM layers of 32 nodes and a Dense layer

To compile a model in keras with TensorFlow, an optimizer (keras.optimizers) has to be chosen, and for this model, Stochastic Gradient Descent (SGD, or gradient descent) has been selected, whose implementation has been widely used in the past with great success in many other projects. This is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The goal is to take small steps repetitively in the direction opposite to the gradient of the function at the current point, because this is assumed to be the direction of the fastest and deepest descent.

This is done until a local minimum is found.

For the loss function, Huber Loss has been chosen, which is the combination of the Gaussian loss function and the Laplace loss function, but it has better performance than the Gaussian loss function. The Huber Loss function is less sensitive to outliers in the data than the squared error loss function. It is also differentiable at 0. It is basically an absolute error, which becomes quadratic when the error is small. How small that error must be to be quadratic depends on a hyperparameter, δ (delta), which can be adjusted in code. The Huber function gets closer to MSE when $\delta \sim 0$ and MAE when $\delta \sim \infty$, with large numbers [6].

In this first Neural model, a Sequential traditional Neural Network was implemented, which is a linear stack of layers in the network. This network configuration (as shown in figure 3) has 1,873 parameters with 4 Dense layers. The result of this simple neural network was very far from optimal, with a result of a loss of 66.45 and an RMSE of 73.21. These results will be kept to work with our hypothesis H1 and H2.

The next step was to build the second network, an LTSM Recurrent Neural Network.

RMSE is in terms of the same units as the dependent variable, in this case, our RUL [14]. This means that there is no absolute good or bad threshold, however it can be defined based on its dependent variable. In this way, theoretical statements about RMSE levels can be made if what is expected of the independent variable in the field of research is already known.

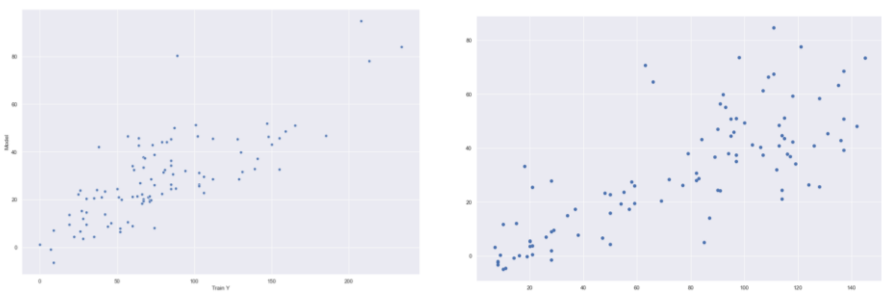


Fig. 5. Plot of RUL of the training data Vs the prediction of the model (left) and Plot of the result of the test data (data that the model has never seen) Vs the prediction of the model (right).

As can be noticed in the code, the model consists of 5 layers, of which 2 are LSTM, a normal layer (Dense), and the output layer, where the output by the mean RUL which is 206 (the mean of the descriptive statistics of the train_FD001 dataset cycles) are multiplied through a lambda function.

TensorFlow's powerful callbacks tool has been used, which provides a great degree of control in many aspects of the model, i.e., which has the advantage of the ability to stop training with this functionality when an expected efficiency is reached, and thus, generating an optimal strategy for the neural model is possible.

By calling to the ***summary()*** function, it can be observed that the total number of parameters is 14,398 divided into the 5 layers of the neural network.

The main strategy in this model is to use the ***LearningRateScheduler()*** function (from tf.keras.callbacks) in the first training of this model to move the learning rate in each epoch, adding a callback to move this learning rate. This will be called in the callback at the end of each epoch. What this does is change the learning rate to a value based on the epoch number. So, in epoch 1, it's 1 times 10 to the -8, times 10 to the 1 in 20. And when the epoch 100 is reached, it will be 1 times 10 to the -8, times 10 to the 5, and so on, and that is 100 over 20. This will happen in every callback because it is set in the callback parameter of the function. An advantage of this strategy is that at the end of the execution, a chart of loss Vs epochs and another chart of loss Vs learning rate can be created. These charts will provide extremely important information about what happens in each epoch and in each step of the training. The ***fit()*** function (***model.fit***) is called, where the strategy is training by 100 epochs in order to later visually analyze the behavior of the learning rate through the epochs. Callbacks is also implemented, to call ***lr_schedule*** as explained above.

Even though the result of the first training ended with a final loss or loss of 42.4795 and an RMSE of 53.2980, it can be observed an incremental trend in the previous graph, although there are also anomalous data that that can be attributed to the type of problem, where there is an infinite number of factors that can produce a great number of anomalous data in the sensors of a turbofan.

To know how much the model fits, the Coefficient of Determination (also called R2, or R squared) is now calculated, which is the proportion of variance (%) in the dependent variable that can be explained by the independent variable [13]. This shows how

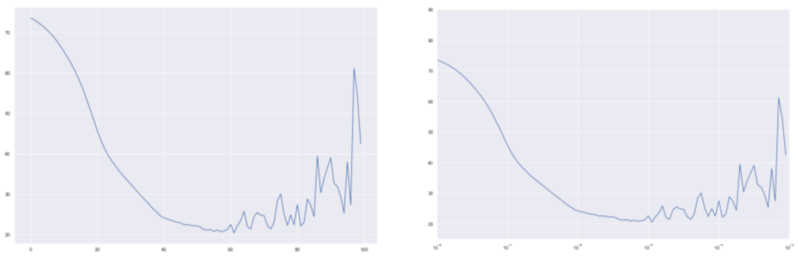


Fig. 6. Charts for Loss Vs Epochs (left) and Loss Vs Learning Rate (right).

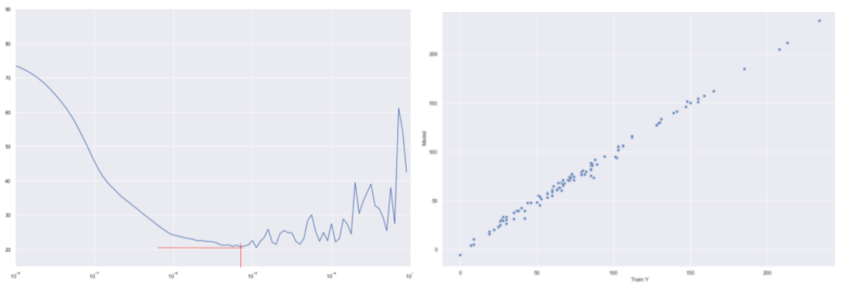


Fig. 7. Plotting two red straight lines on our Loss Vs Learning Rate graph (left) and Plot of RUL of our training data Vs the prediction of our model after updating the learning rate and using ***myCallback*** and ***on_epoch_end*** (right).

close the data is to the fitted regression line. The model output with the test data (which the model has never seen) was -0.602 of R2, which states that the model needs to be improved.

This result when calculating RMSE was 52.60, this means that on average there could be a deviation of the RUL result of 52.60 units up or down. Now observing what the model did in each epoch is needed, and how the loss changed during the training. For doing this, Loss Vs epochs is plotted, and with this a notion of what was happening with the loss will be revealed, if it was going down and if it behaved erratically at some point, in order to adjust our variables at the right moment. The history variable is used, which is created by assigning the history of the call to the ***model.fit***() function, thanks to history, access to all the information about what was happening at each moment in our training is possible.

Having this information on how loss has behaved during the epochs, it can be observed that it was going well during the first 50 epochs, when the loss variable began to rise, which was not efficient at all. Next step is looking at the chart of the right, Loss Vs Learning Rate, to get more information about the behavior of the learning rate and see if making a decision with this information is possible.

By visually observing in the graph at which moment the loss variable begins to have an erratic increasing behavior, the learning rate of the exact moment where this changes took place can be observed, to use it as the initial learning rate in Gradient Descent and

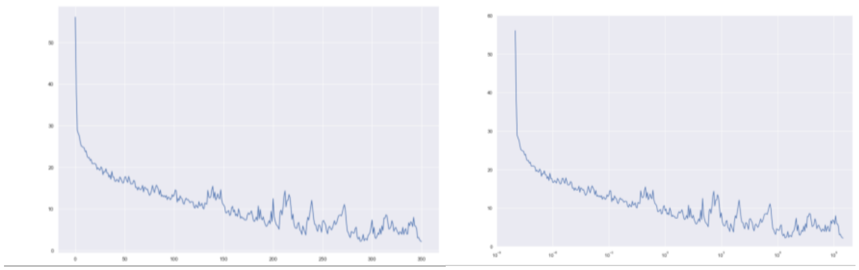


Fig. 8. Charts for Loss Vs Epochs (left) and Loss Vs Learning Rate (right).

run the whole model again. To do this, two ***red*** straight lines are drawn from the moment the learning rate began to behave erratically.

As can be seen in the previous chart, the learning rate 7 * 10^{-6} has been chosen to use it in the second training and see how the model behaves, doing the same tests and thus the same graphs after the training can be observed to be able to make other decisions.

To prepare the second training, the learning rate scheduler is no longer needed, since it has fulfilled its objective in the first training. In this training SGD will do its job, but now with an optimal initial learning rate.

A class ***myCallback*** is now created, in which the function ***on_epoch_end()*** will be defined, which will be in charge of stopping the training when RMSE reaches a target value, in this case 0.5. This is done to avoid that the training passes through optimal values and behaves erratically, returning to inefficient large values.

The call to the fit function (***model.fit***) can be executed next, but now, training for 500 epochs, as intensive training, and also calling ***callbacks*** function is needed (to ***myCallback*** class), to stop the training when reaching an RMSE less than 0.5.
When the model was executed again with ***model.fit*** using the new strategy, the result of the training with 500 epochs was a loss of 4.3183, which is quite good. But the expectation was to see if a better value could be given, so another execution of 500 epochs was performed, this time with ***callbacks*** to stop when RMSE is less than 3.5 and loss is approximately 2.

This strategy worked pretty well, ***callbacks*** function helped so that the behavior of the model did not raise the loss again, obtaining a final loss of 2.1686, which is much better than the previous one.

The right image in figure 7 shows a perfect line in graphing the training RUL and the result of the model predictions.

Next step was to observe how the model behaved with data that it had never seen, after knowing that the behavior of the model with the training data has been optimal.

The behavior with the test data that the model had never seen was observed, and with this, an R2 of 0.701 and an RMSE of 20.419 were reached. This result is quite good for this type of neural models with time series.

As it was shown in previous runs, it was possible to observe what the model did in each epoch and how the loss changed during training. And for this a graph of Loss Vs epochs was created, and with this, a good insight of what was happening with the loss was provided, if it was going down and if it behaved erratically at some point, to adjust our variables at the right time.

Analyzing Loss Vs Epochs chart can help to observe behaviors through the epochs in training, the algorithm arrived at a Loss that tends close to zero, which is ideal for the model, and analyzing Loss Vs Learning Rate graph provides information to conclude that a model with which a model can reliably calculate RUL in turbofan units in the aerospace industry was achieved.

4 Results of Experiments

H1 has been accepted when creating the models in figure 3 and figure 4 and iterating through different strategies in each experiment. The traditional model in figure 3 threw results very far from an optimal model and using LTSM neural network layers reached an optimal model. H1 was accepted because when a traditional and sequential Neural Network was used the loss result was 66.45 and RMSE was 73.21, and when using LTSM the loss was 2.1686 and an RMSE of 3.5. H2 has been rejected when analyzing these results.

5 Conclusions and Future Research

With the model and adjustments made in this work, optimal results for a model of this type were achieved, where the main characteristic of the model is that the estimation of the remaining useful life from data such as those of multiple sensors in a turbine, and where this engine is it degrades through time, it can be considered as the learning of a regression function, which maps a multivariate time series to a real value number. Challenges in the supervised learning-based approach have been highlighted, such as missing data in datasets, learning to estimate RUL values, sensor noise, anomalies due to lack of knowledge of the beginning of engine degradation, and implementation of neural models with very large multivariate time series, dealing with multiple operating conditions, etc. A solution strategy has been executed in the context of these challenges.

It is important to highlight that having such powerful tools as TensorFlow, Keras, Numpy and Pandas provide the programmer with the ability to create strategies with neural networks as powerful as observing the learning rate and loss during the epochs in a visual way, generating great positive impacts on the artificial intelligence industry, such as accelerating strategies to reach exponentially better results faster.

In the United States, in collaboration with NASA through multiple aviation contracts, the company Ampaire, Inc. is validating the capabilities of electric aircrafts. Such systems are starting to use RTOS operating system have the advantage of being really "in real time" (which, so far, do not exist similar systems in aviation). Advanced aviation systems such as Ampaire's, could be implementing their sensor strategy with a microcontroller working with RTOS, such as the product of the company Hover, Inc., which in partnership with the company Pinnacle Aerospace, Inc., have created the first Hardware under RTOS for aviation systems, certified by the FAA. This type of systems could be adapted to work with turbofan units and could take advantage of high availability architectures for decision making in monitoring systems, where, besides the great potential provided by Machine Learning solutions, they can implement a reliable approach in the sensor reading strategy that combines algorithms for reading sensors, preprocessing data with inferential statistics, and producing a high decision-making capacity, with less computing resources.

References

1. Chao M. C., Kulkarni, Goebel K., and Fink O. (2020). Aircraft Engine Run-to-Failure Dataset under real flight conditions, NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, CA, USA.
2. Eide I., Westad F. (2018), "Automated multivariate analysis of multisensor data submitted online: Real-time environmental monitoring". PLoS ONE 13(1): e0189443. https://doi.org/10.1371/journal.pone.0189443.
3. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735–1780.
4. Jansohn P. (2013), "Overview of gas turbine types and applications". Modern Gas Turbine Systems. High Efficiency, Low Emission, Fuel Flexible Power Generation. Paul Scherrer Institute, Switzerland.
5. Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. In ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 89–94.
6. Mayer G. P. (2019). An Alternative Probabilistic Interpretation of the Huber Loss, Uber Advanced Technologies Group. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. USA.
7. Nguyen, K.T.P., Medjaher K. (2019), "A new dynamic predictive maintenance framework using deep learning for failure prognostics". Reliability Engineering and System Safety, 188. 251−262. ISSN 0951-8320.
8. Olah Ch. (2015, Aug 27), Understanding LSTM Networks, Recurrent Neural Networks, CA, USA. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
9. Petersen R., Khan O. (2019), Aviation Maintenance Technician Certification Series, Propulsion, Module 14 for B Certification. EASA, European Aviation Safety Agency. Aircraft Technical Book Company, CO, USA.
10. Shane T. (2009), "Systematic Sensor Selection Strategy for Turbofan Engine Diagnostics". GRC Propulsion Control and Diagnostics Workshop, NASA, QinetiQ North America, USA.
11. TV V., Gupta P., Malhotra P., Vig L., Shroff G. (2018), "Recurrent Neural Networks for Online Remaining Useful Life Estimation in Ion Mill Etching System". Annual Conference of the PHM Society, 10(1), Sept.
12. Mendez R., Ochoa A., Zayas B., Perez M., Quintero O. (2020), "Implementation of Big Data in Intelligent Analysis of Data from a Cluster of ROVs Associated with System of Prevention and Reparation of Hydrocarbon Leaks to Optimize their Distribution in Gulf of Mexico". Research in Computing Science, 149(7), pp. 73−85, México.
13. Reyes V., Cossio E., Pescador A. (2018), "Demand Forecasting Applied to Radio Frequency Identification Technology". Research in Computing Science, 149(7), pp. 97−109, México.

14. Hernandez R., Gabbasov R., Suárez J. (2015), "Improving Performance of Particle Tracking Velocimetry Analysis with Artificial Neural Networks and Graphics Processing Units". Research in Computing Science, 104, pp. 71−79, México.

ISSN 1870-4069

Feature Subset Selection in Electroencephalographic Signals Using Typical Testors

Alexis Gallegos, Dolores Torres, Aurora Torres, Eunice Ponce de León

Benémerita Universidad Autónoma de Aguascalientes, Aguascalientes, Mexico

alexisedm@gmail.com
{mdtorres, atorres, eponce}@correo.uaa.mx

Abstract. Motor imagery (MI) is a mental representation of movement without performing or tensing any muscles. MI requires a conscious activation of the same brain regions involved in actual movement. Brain signals have been explored for multiple applications in biomedical engineering, such as the development of brain-computer interfaces (BCI). BCI systems are designed to translate users' intentions into control signals, commands, or codes. Nevertheless, the major challenge in BCI system development is classifying MI signals recorded by an electroencephalogram (EEG). This paper focuses on applying the testor theory and the logical combinatorial pattern recognition approach for feature selection to reduce the feature representation space for classification tasks. The EMOTIV EPOC+ EEG device recorded the MI-EEG signals with 14 electrodes.

Keywords. Typical testors, feature subset selection, electroencephalographic signals, motor imagery

1 Introduction

This paper focuses on the application of testor theory and the logical combinatorial pattern recognition approach for feature selection. The problem of selecting the subset of features that best describes a phenomenon from a larger set allows to reduce the size of solution space, so that results close to the optimum or the optimum itself are obtained with less resources (time and memory) [1].

Therefore, the aim of the paper was to reduce the MI-EEG (Motor Imagery Electroencephalographic) signals feature representation space for classification tasks. These signals were recorded by the Emotiv EPOC+ device which describes the signals by means of 14 electrodes distributed over the scalp.

The EEG signals represent the electrical brain activity created by billions of neurons [2]. This activity represents the communication between the body and the brain. The analysis of EEG signals is highly relevant in health research for diagnosis, treatment, and monitoring of different diseases [2], [3].

On the other hand, motor imagery, or MI, is a mental performance of movement without any physical activation. The movements analyzed were opening and closing of the hand. The practice of MI is used in the context of sports as well as in rehabilitation treatments because it requires the activation of the same brain areas. It has also shown positive results in learning of physical skills and strength gain [4].

The present document is composed of three more sections. The second section describes the basic concepts used: typical testors, electroencephalographic signals and motor imagery. Section 3 provides the framework that allowed for the typical testors analysis. The framework includes the recording of MI-EEG signals taken with the support of six test subjects, the preprocessing of the data and the selection of the minimal subset of feature that will allow for a correct classification.

Finally, section 4 provides the typical testors found, which represent the minimum subset of features for describing objects. In addition, the informational weight (IW), which represents a measure of significance for each feature involved, is described. Thus, the higher its value, the more determinant it is to differentiate classes of objects.

2 Important Concepts

2.1 Typical Testors

Technological advances have led to the generation of large amounts of data at an unprecedented speed. These data describe objects or phenomena with high number of features, resulting in a challenge for machine learning and data mining research [5]. Feature Selection is the area of pattern recognition *"responsible for identifying those features that provide relevant information for classification purposes"* [6].

There are different tools or methodologies applied in addressing feature selection. One of them is the logical combinatorial pattern recognition approach, where testor theory is applied for such task [6]–[8].

Testor theory was introduced in the 60s to locate faults in electronic circuits [7].

Dimitriev, Zhuravlev and Krendeleiev's testor theory approach [9] for classification and feature selection establishes that classes are disjoint sets, the criterion of comparison between features is Boolean, and the criterion of similarity between objects accepts that two objects are different if at least one of their features is also different [10].

Testor theory defines a testor as a feature subset that does not confuse object belonging to different classes. I.e., no object belonging to class T_0 can be confused with any object of class T_1 according to the values in its features [11].

According to Shulcloper et al. [12], the definition of testor can be extended to more than two classes.

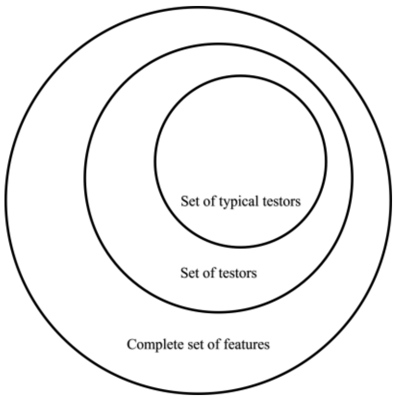


Fig. 1. Sets of features, testors and typical testors [8].

Within the set of testors there is the set of typical testors (see Fig. 1), also known as irreducible testors. A testor can be a typical testor if by eliminating one of its features, the remaining subset is no longer a testor. Therefore, a typical testor is the minimum feature subset needed to distinguish objects of different classes [10], [12].

The importance of the typical testor calculation lies in the reduction of the feature representation space, feature selection, as support for classification and pattern recognition systems [10].

As described so far, the main aim of the testor theory is feature selection. However, testor theory can be used to determine the relevance of each feature by computing the informational weight from the set of typical testors [13]. The informational weight is calculated by means of the relative frequency. Let be τ the number of typical testors found and $\tau(i)$ the number of typical testors in which the feature x_i appears, the informational weight is given by [8]:

$$P(x_i) = \tau(i) / \tau. \tag{1}$$

The obtained score represents a measure of significance for each feature. This means that the higher the score of the feature, the greater its relevance in class distinction [8], [14], [15].

2.2 Electroencephalographic Signals

Brain is a complex part of the human body which plays an important role for controlling behavior of human body according to different stimuli. The study of the functional and cognitive behavior of the human brain has been an important area of medical research to find better diagnoses and treatments for brain related issues [3]. These studies can be performed by processing electrical brain activity, specially through computational modeling. The electrical brain activity is created by billions of interconnected neurons across different areas of the brain [16]. These neurons *"act as information carriers between the body and brain"* [3]. Voltage potential resulting from current flow in and around neurons can be recorded by electrodes placed on the scalp and reported as electroencephalographic (EEG) signals [16], [17].

As a definition, *"electroencephalography (EEG) is the non-invasive measurement of the brain's electric fields"* [17]. According to Keenan et al. [18], the EEG recording allows for data analysis from the frequency and amplitude domains, i.e., time and voltage. EEG signals are complex because they correspond to a mixture of information, physiological artifacts (eye movements, muscle movements, heartbeats, sweat) or technical artefacts (power supply line, electrode disconnection) [19].

Electroencephalography has applications in several domains such as health, education and, entertainment [16]. For example, EEG is decisive for in the diagnosis, treatment and monitoring of epileptic syndrome patients, and the study of sleep patterns, depth of anesthesia, and attention deficit hyperactivity disorder. There are also applications in cognitive and affective monitoring such as level of fatigue, mental workload, mood, or emotions, even stress control [16], [20], [21]. Outside the medical field, there are also applications such as BCI systems that allow the translation of EEG signal patterns into messages or commands for applications or interactive devices [16] with the aim of making human computer interaction more natural, especially for people with neuro-muscular disabilities [22].

2.3 Motor Imagery

Motor imagery (MI) is defined by Mokiento et. al. [23] as the mental performance of movements without by any kind of peripheral muscular activity. I.e., MI is a mental representation of movement without any body movement [4], e.g., opening or closing the left or right hand without executing it [24]. The practice of MI requires the conscious activation of the same brain regions involved in the preparation and execution of movement. In addition, MI allows motor development and the learning of motor skills, including the gain of strength in specific muscle groups [4], [25].

MI has been applied in the sports context and positive effects have been reported in speed, performance accuracy, muscle strength, movements dynamics and motor skill performance. From the medical point of view, there have been positive results in rehabilitation in patients with neurological conditions, e.g., stroke, spinal cord injury, or Parkinson disease [4].

Computer science has studied motor imagery for the development of brain computer interface (BCI). A BCI system allows the communication between the brain and external devices without the involvement of peripheral nerves or muscles [26]. The most important BCI systems are the MI-based BCI systems, which involves real-time applications such as contactless writing, prosthetic arms, virtual reality systems, gamming apps, wheelchairs, etc.[27].

The major challenge in the design of MI-based BCI systems is the classification of EEG signal due to events such as eye blink, eye movement, muscular movements, teeth grinding, and heart rhythm interfere with the EEG signal recording resulting in a noisy signal [28]. Consequently, multiple frameworks have been developed in the literature with the aim of making EEG signal processing more efficient [28], [29].

3 Framework

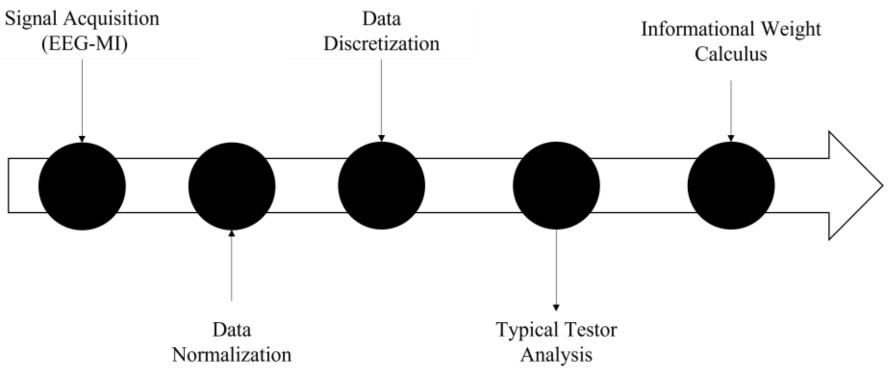


Fig. 2. Framework for typical testors analysis.

This section of the document describes the framework used to perform the typical testors analysis applied to EEG signals with motor imagery. This framework aims to select a feature subset thar allows the correct discrimination of signals corresponding to the motor imagery of "open" and "close" the right hand.

As shown in Figure 2, the process began with the EEG signal recording by means of the Emotiv EPOC+ EEG device with 14 channels (AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4), distributed according to the 10-20 electrode placement system, and a sampling rate of 128 samples per second.

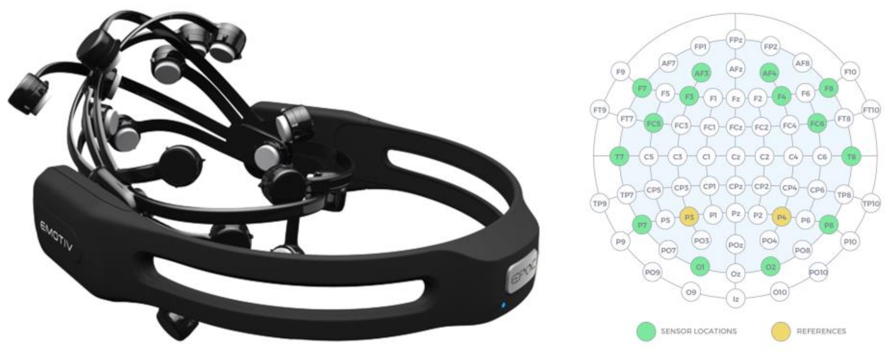


Fig. 3. Emotiv EPOC+ device, 14 signals and 10-20 system [30].

Six test subjects were used for signal sampling. For each one, six five-second samples were taken: three correspond to the intention to "open" and three more to "close" the hand. Additionally, the open-source application Cykit was used to record EEG signals in csv files, which were used as parameters for the following phases of the methodology.

The second phase of the framework consisted of normalizing the data set since, although the fourteen features describe a brain electrical signal, there are small variations that may cause some features to be dominated by others. To avoid this issue, every feature was standardized using z-score normalization. In this way, the set of features has the same scale.

The EEG signals are recorded with continuous values; therefore, the third phase of the framework was to apply a discretization. This, in addition, with the objective of making the typical testors analysis easier.

Finally, the phases four and five properly involve a feature subset selection (FSS) process using the logical combinatorial approach by means of typical testor analysis (see section 2.1). This process used in data mining provides tools for the efficient reduction of the number of features describing objects, with the purpose of removing irrelevant features resulting in more stable representations.

The results obtained from the application of the framework described above are shown in section 4 below.

4 Results and Conclusions

As a result of phase 1, a database of 23,846 records described by the signals from the 14 electrodes of the Emotiv device was obtained. Specifically, the database included 12,190 records corresponding to the intention to open the hand and 11,656 to the intention to close the hand.

As mentioned in the previous section, the data set was preprocessed through a normalization and discretization process. Once completed, a random sampling of 500 records per class was performed, ending the preprocessing a smaller data set that will be the parameter for the feature subset selection application, i.e., the computing of typical testors.

Table 1. Testors and Typical Testors

Typical Testors													
AF3	**F7**	**F3**	**FC5**	**T7**	**P7**	**O1**	**O2**	**P8**	**T8**	**FC6**	**F4**	**F8**	**AF4**
1	1	1	1	1	1	1	1	1	1	1	0	1	1

The typical testors were calculated by means of a Python library of the exhaustive method using the Python library TestoresTipicos.py developed by Daniel Barajas, PhD student at Autonomous University of Aguascalientes, Mexico. The testors and typical testors found are shown in Table 1. As can be seen, the typical testors are represented as binary values where zero represents the absence of the feature, i.e., it does not provide relevant information and one means that the feature provides essential information.

For this study, two testors were obtained, one of which is a typical testor. This typical testor is the minimum feature subset needed to distinguish objects of different classes ("open" and "close"). This typical testor is used to calculate the informational weight of each feature as described in section 2.1.

Table 2. Informational Weight

Feature	IW	Feature	IW
AF3	100%	**O2**	100%
F7	100%	**P8**	100%
F3	100%	**T8**	100%
FC5	100%	**FC6**	100%
T7	100%	**F4**	0%
P7	100%	**F8**	100%
O1	100%	**AF4**	100%

Table 2 shows the informational weight calculated from the typical testor resulting from Table 1. Each percentage represents a measure of significance for each feature involved. In this sence, 13 of the 14 features are essential (100% of informational weight) to defferentiate the two classses, while the feature F4 with 0% of informational weight presents no relevant information for this process.

In this manner, the objective of reducing the dimensionality of the problem is achieved by describing the clases with a smaller number of features, making the representation of objects easier and as a support for classification systems.

As future work, it is proposed to test the typical testor found by evaluating the performance of different machine learning algorithms sucha as artificial neuronal networks or support vector machines.

References

[1] M. D. Torres Soto, A. Torres Soto, and E. Ponce de León Sentí, “Algoritmo Genético y Testores Típicos en el Problema de Selección de Subconjuntos de Características.” 2006, [Online]. Available: http://www.iiisci.org/journal/CV$/risci/pdfs/C415DR.pdf.

[2] W. H. Wolberg, W. N. Street, D. M. Heisey, and O. L. Mangasarian, "Computer-derived nuclear features distinguish malignant from benign breast cytology," *Hum. Pathol.*, vol. 26, no. 7, pp. 792–796, 1995, doi: http://dx.doi.org/10.1016/0046-8177(95)90229-5.

[3] J. S. Kumar and P. Bhuvaneswari, "Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study," *Procedia Eng.*, vol. 38, 2012, doi: 10.1016/j.proeng.2012.06.298.

[4] R. Dickstein and J. E. Deutsch, "Motor Imagery in Physical Therapist Practice," *Phys. Ther.*, vol. 87, no. 7, pp. 942–953, Jul. 2007, doi: 10.2522/ptj.20060331.

[5] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, Jul. 2018, doi: 10.1016/j.neucom.2017.11.077.

[6] V. I. González Guevara, S. Godoy Calderon, E. Alba Cabrera, and J. Ibarra Fiallo, "A Mixed Learning Strategy for Finding Typical Testors in Large Datasets," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 1st ed., Montevideo, Uruguay: Springer, 2015, pp. 716–724.

[7] E. Alba-Cabrera, R. Santana, A. Ochoa, and M. Lazo-Cortés, "Finding typical testors by using an evolutionary strategy," *Proc. Fith Ibero Am. Symp. Pattern Recognit.*, 2000.

[8] A. E. Gallegos Acosta, F. J. Álvarez Rodríguez, M. D. Torres Soto, and A. Torres Soto, "Identificación de factores de riesgo en patologías médicas mediante métodos de selección de subconjuntos de características [recurso electrónico]," Universidad Autónoma de Aguascalientes, Aguascalientes, México, 2018.

[9] A. N. Dmitriev, Y. I. Zhuravlev, and F. P. Krendeleiev, "On the mathematical principles of patterns and phenomena classi-" cation," *Diskretn. Anal*, vol. 7, no. 3, p. 15, 1966.

[10] Y. Santiesteban Alganza and A. Pons Porrata, "Lex: Un nuevo algoritmo para el calculo de los testores tipicos," *Ciencias Mat.*, vol. 21, no. 1, Jan. 2003, [Online]. Available: https://dibpxy.uaa.mx/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsgii&AN=edsgcl.147016161&lang=es&site=eds-live&scope=site.

[11] J. Ochoa Somuano, "Técnicas de Selección de Atributos para la Categorización Automática de Escenas Visuales," Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, 2005.

[12] J. Ruíz Shucloper, E. Alba Cabrera, and M. Lazo Cortés, "Introducción a la Teoría de Testores." Departamento de Ingeniería Electrica, CINVESTAV-IPN, p. 197, 1995.

[13] G. Díaz Sánchez *et al.*, "Typical Testors Generation Based on an Evolutionary Algorithm," *Intell. Data Eng. Autom. Learn. - IDEAL 2011*, vol. 6936, Sep. 2011, doi: 10.1007/978-3-642-23878-9.

[14] J. A. Santos, A. Carrasco, and J. F. Martínez, "Feature Selection using Typical Testors applied to Estimation of Stellar Parameters," *Computación y Sistemas*, vol. 8. Instituto Politécnico Nacional, México, pp. 15–23, 2004.

[15] M. D. Torres, E. Ponce de León, C. A. Ochoa, A. Torres, and E. Díaz, "Mecanismos de Aceleración en Selección de Características Basada en el Peso Informacional de las Variables para Aprendizaje no Supervisado," *Revista Iberoamericana de Sistemas, Cibernética e Informática*, vol. 6. Revista Iberoamericana de Sistemas, Cibernética e Informática, pp. 29–34, 2009.

[16] G. Coro, G. Masetti, P. Bonhoeffer, and M. Betcher, "Distinguishing Violinists and Pianists Based on Their Brain Signals," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11727 LNCS, pp. 123–137, doi: 10.1007/978-3-030-30487-4_11.

[17] A. Biasiucci, B. Franceschiello, and M. M. Murray, "Electroencephalography," *Curr. Biol.*, vol. 29, no. 3, Feb. 2019, doi: 10.1016/j.cub.2018.11.052.

[18] S. A. Keenan, O. Carrillo, and H. Casseres, "Electroencephalography," in *Encyclopedia of Sleep*, Elsevier, 2013.

[19] A. Santillan Guzman, H. Z. Ramírez Uriarte, J. J. Oliveros Oliveros, M. M. Morin Castillo, and H. Ramírez Díaz, "Interfaz Gráfica Intuitiva para el Procesamiento de Señales EEG," *Memorias DEl Congr. Nac. Ing. Biomédica*, vol. 5, no. 1, pp. 146–149, 2018, doi: dx.doi.org/10.24254/CNIB.18.20.

[20] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: a systematic review," *J. Neural Eng.*, vol. 16, no. 5, p. 51001, Aug. 2019, doi: 10.1088/1741-2552/ab260c.

[21] F. Ramos Argüelles, G. Morales, S. Egozcue, R. M. Pabón, and M. T. Alonso, "Técnicas de electroencefalografía: principios y aplicaciones clínicas," *An. Sist. Sanit. Navar.*, vol.

32, pp. 69–82, Jun. 2009, [Online]. Available: http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1137-66272009000600006&nrm=iso.

[22] V. P. Oikonomou, K. Georgiadis, G. Liaros, S. Nikolopoulos, and I. Kompatsiaris, "A Comparison Study on EEG Signal Processing Techniques Using Motor Imagery EEG Data," in *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, Jun. 2017, doi: 10.1109/CBMS.2017.113.

[23] O. A. Mokienko, L. A. Chernikova, A. A. Frolov, and P. D. Bobrov, "Motor Imagery and Its Practical Application," *Neurosci. Behav. Physiol.*, vol. 44, no. 5, Jun. 2014, doi: 10.1007/s11055-014-9937-y.

[24] Z. Tayeb *et al.*, "Validating Deep Neural Networks for Online Decoding of Motor Imagery Movements from EEG Signals," *Sensors*, vol. 19, no. 1, Jan. 2019, doi: 10.3390/s19010210.

[25] T. Mulder, "Motor imagery and action observation: cognitive tools for rehabilitation," *J. Neural Transm.*, vol. 114, no. 10, Oct. 2007, doi: 10.1007/s00702-007-0763-z.

[26] S. Kumar and A. Sharma, "A new parameter tuning approach for enhanced motor imagery EEG signal classification," *Med. Biol. Eng. Comput.*, vol. 56, no. 10, Oct. 2018, doi: 10.1007/s11517-018-1821-4.

[27] S. R. Sreeja and D. Samanta, "Classification of multiclass motor imagery EEG signal using sparsity approach," *Neurocomputing*, vol. 368, Nov. 2019, doi: 10.1016/j.neucom.2019.08.037.

[28] S. Kumar, A. Sharma, K. Mamun, and T. Tsunoda, "A Deep Learning Approach for Motor Imagery EEG Signal Classification," in *2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, Dec. 2016, doi: 10.1109/APWC-on-CSE.2016.017.

[29] S. R. Sreeja, J. Rabha, K. Y. Nagarjuna, D. Samanta, P. Mitra, and M. Sarma, "Motor Imagery EEG Signal Processing and Classification Using Machine Learning Approach," in *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, Oct. 2017, doi: 10.1109/ICTCS.2017.15.

[30] J. Gwizdka, R. Hosseini, M. Cole, and S. Wang, "Temporal dynamics of eye-tracking and EEG during reading and relevance decisions," *J. Assoc. Inf. Sci. Technol.*, vol. 68, Aug. 2017, doi: 10.1002/asi.23904.

Electronic edition
Available online: http://www.rcs.cic.ipn.mx