PUC-Rio

BioSleep

version 1.2

Autor:
Maria Leandra Guateque Jaramillo

Advisor:
 Sérgio Lifschitz

REVISION HISTORY

DATE	VERSION	DESCRIPTION	AUTOR
15/07/2022	1.0	Initial Version	Maria Leandra
16/08/2022	1.1	Modules inclusion	Maria Leandra
28/08/2022	1.2	Final Version	Maria Leandra

TABLE OF CONTENTS

BioSleep

1. Introduction

1.1. Scope of the topic

The system provides an easy-to-use interface for analyzing EEG signals and identifying the presence of events associated with sleep disorders (apnea, hypopnea and respiratory effort-related arousals (RERA)). One of the more well-studied sleep disorders is Obstructive Sleep Apnea Hypopnea Syndrome (or simply, apnea). Apneas are characterized by a complete collapse of the airway, leading to awakening, and consequent disturbances of sleep. While apneas are arguably the best understood of sleep disturbances, they are not the only cause of disturbance. Sleep arousals can also be spontaneous, resulting from teeth grinding, partial airway obstructions, or even snoring.

1.2. Objective

This document serves as an input to technical design decisions, coding, and provides additional input for planning system and acceptance tests.

1.3. Definitions and Abbreviations

1.3.1. Definitions

- *Mat*: are files that are in the binary data container format that the MATLAB program uses. The extension was developed by Mathworks and MAT files are categorized as data files that include variables, functions, matrices and other information. This case is a Matlab V4 file containing the PSG signal data.
- *Hea:* record header file - a text file which describes the format of the signal data.
- *Arousal:* arousal and sleep stage annotations, in WFDB annotation format.

- *-arousal.mat:* a Matlab V7 structure containing a sample-wise vector with three distinct values (+1, 0, -1) where:
  - +1: Designates arousal regions
  - 0: Designates non-arousal regions
  - -1: Designates regions that will not be scored

1.3.2. Abbreviations

- **AASM**- - American Academy of Sleep Medicine
- **AUROC**- - Area Under the Receiver Operating Characteristic
- **AUPRC**- - Area Under the Precision-Recall Curve
- **CCNL**- - Computational Clinical Neurophysiology Laboratory
- **CNN**- - Convolutional Neural Networks
- **EEG**- - Electroencephalography
- **FP**- - False Positives
- **PSG**- - Polissonography
- **RERA**- - Respiratory Effort-Related Arousal
- **tanh**- - Tangente Hiperbólica
- **TN**- - True Negatives
- **TP**- - True Positives

1.4. Overview

In section 2 the functional and non-functional requirements will be detailed. Section 3 sets out guidelines for verifying and validating software artifacts. Section 4 presents the main components of the solution: database used, architecture of the deep neural network that will be used to identify RERA events and the PySimpleGUI tool. Section 5 details the architecture of the project and its organization. Section 6 presents all the tests performed to control the quality of the program, the test cases and the reports obtained. Finally, in section 7 the user guide is presented.

2. Requirements

2.1. Functional Requirements

[FR1] Access to the system
The system must allow users access to using their user and password. The user enters his user data (nickname and password).
[FR2] Recover password
The system must allow users to recover passwords through their email. The system will send a message to the registered email with the password registered.
[FR3] Data loader and formats support
The system must allow users to search and upload files with the analysis of a certain patient. The system must accept upload 3 files in the following formats: .mat, .arousal and .hea.
[FR4] View files
The system must show users the list of files that have been uploaded, with basic patient information such as ID.
[FR5] Evaluation of biosignals and issuance of alerts
The system must generate alerts based on the analysis of the signals using a pre-trained convolutional neural network which takes 6 EEG signals as input: 2 frontal leads, 2 occipital leads, and 2 central leads, according to what is established by the AASM.
[FR6] Results viewer
The system must display the following data on a user interface: total apnea/hypopnea episodes found, total RERA events found, AUROC and AUPRC values obtained by evaluating the patient's records.

2.2. Non-Functional Requirements

2.2.1. Security

[NFR1] Access restriction
The system must restrict access to the application to only registered users.
[NFR2] Database access

Only the development team will be able to access the local server and the software (application).

2.2.2. Usability

[NFR3] Use of a graphical interface
The system must show a graphical interface able to interact with a user. The interface must be developed in a python environment that contains the necessary packages as stated in the file Requirements.txt.
[NFR4] Use the PUC-Río logo

2.2.3. Reliability

[NFR5] Data persistence
The system must allow restoring the user session when there is a blunt interruption.

2.2.4. Performance

[NFR5] Display the result
The system should display the query result in the application in a maximum of approximately 20 minutes (reference time in signal records of up to 8 hours in duration).

2.2.5. Availability

[NFR5] Support using the tool on the computer.

2.2.6. Data and Storage

[NFR6] Support structured data model
Due to the amount of data that will compose the database that supports the tool, a relational database with SQL must be set up.

3.   Validation and Verification

The software specification and architecture requirements will be subjected to validation (to identify if we are building the right product) and verification (to identify if we are building the product). Validation and static verification allow the identification of errors and specification defects. For example, functional requirements verification can be used to detect whether memos are consistent with each other and correctly specified. The eventual software failures of the tools and data load will be dealt with in the planning and execution of the system tests.

4.   Solutions Components

4.1.   Physionet Database

The data set used was *The PhysioNet Computing in Cardiology Challenge 2018* , whose data was provided by the Massachusetts General Hospital Computational Clinical Neurophysiology Laboratory (MGH/CCNL) and the Data Animation Laboratory (CDAC). It has polysomnographic records (PSG) of 1,985 individuals who were monitored in an MGH sleep laboratory for one night. The database is partitioned into two groups: (i) 994 records for balanced training and (ii) 989 records for Test (whose classes are not in the public domain). Each of the polysomnographic records has a total of 13 physiological signals regarding the measurement of brain, muscle, cardiac and respiratory activity.

The PSG records are divided into 30-second intervals, each interval has different types of observations (annotations) made by MGH certified sleep technologists associated with the following categories: (i) sleep stages, (ii) Awakenings related to respiratory events and (iii) awakenings related to movement disorders such as bruxism and restless legs syndrome. As for the excitations, these were classified according to their type in the categories: spontaneous excitations, RERA, bruxisms, hyperventilations, hypo-apneas, apneas (central, obstructive and mixed), vocalizations, snoring, periodic leg movements

(PLM), periodic breathing of Cheyne-Stokes or partial airway obstructions.

OBS: Only the training set records are considered because the supervised learning method is used. Then, from each of the PSG records, the signals of electroencephalography (EEG) were extracted along with the class labels divided into: no events, RERA Event and Events associated with apnea/hypopnea.

4.2. Deep Neural Network

The network architecture for evaluating EEG signals, formed by three convolutional blocks (convolutional and Max-pooling layers), as illustrated in Figure 1.
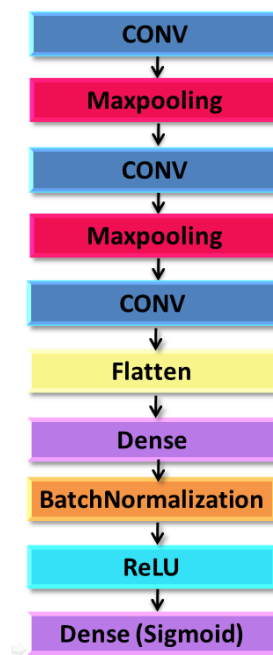

Figure 1. Convolutional Neural Network for evaluated EEG signals for BioSleep System.

This architecture is inspired by the convolutional model[1], which was used to extract characteristics from the EEG signals for the classification of sleep stages. The proposed architecture uses as functions of non-linear activation the function ReLu for the

---

[1] "SLEEPNET: Automated Sleep Staging System via Deep Learning." 26 Jul. 2017, https://arxiv.org/abs/1707.08262. Se consultó el 17 agoust. 2022.

convolutive blocks, and the sigmoid function for the classification layer. In addition, a BatchNormalization layer was added to the end of the architecture, followed by a ReLU layer, as its use allows reducing the change in covariance, allowing each layer of the network to learn more independently of the others, in addition to to smooth out the objective function and improve the performance of the network.

4.3.   PythonSimpleGUI

Is a python library that wraps tkinter, Qt (pyside2), wxPython and Remi (for browser support), allowing very fast and simple-to-learn GUI programming. PySimpleGUI defaults to using tkinter, but the user can change to another supported GUI library by just changing one line.

It is completely based on the Python language and is used to develop GUI interfaces very conveniently. It has a very friendly Python-style interface that substantially improves development efficiency. Its main features include:

- The interface windows and controls used are the same as for tkinter, Qt, WxPython, and Remi.
- The written code is reduced between 50% and 90%.
- You don't need to write the callback function.
- All controls can be accessed under the GUI.
- Supports web and desktop GUI.
- It has a friendly interface
- It is geared towards novice and experienced Python developers.
- There are more than 170 demo programs that explain how to integrate currently popular packages, such as OpenCV, Matplotlib, PyGame, etc.
- There is extensive documentation.

The features of PySimpleGUI can be divided into "high level" calls that allow the user to perform input/output operations in a single function call and "custom GUI" capabilities. High level calls usually start with "Popup". The most basic is Popup, which displays a variable number of items in a popup window. They resemble the built-in "print" statement, taking any number of variables in any format and displaying them in a window. Popup calls that collect user input include PopupGetText, PopupGetFile, and PopupGetFolder. Custom

GUIs are achieved by first creating a "GUI layout" that is used to create and display a window. The GUI layout is made up of "Elements", PySimpleGUI's term for a GUI "Widget" [2]. Table 1 shows some of the elements available in PySimpleGUI.

Table 1. Summary elements availables in PySimpleGUI

UI Elements		Bottons		Abbreviations
Checkbox Column FileBrowse Image InputCombo Listbox Multiline Output	ProgressBar Radio ReadFormButton RealtimeButton SimpleButton Slider Spin Text	FolderBrowse Cancel Exit FileBrowse FileSaveAs FolderBrowse No OK ok	Quit ReadFromButton RealtimeButton Save SimpleButton Submit Yes	T= Text Txt = Text In= InputText Combo = InputCombo DropDown = InputCombo Drop = InputCombo

---

[2] (n.d.). PySimpleGUI. Se recuperó el agosto 24, 2022 de https://www.pysimplegui.org/

5. Project Architecture

5.1. Conceptual view of data
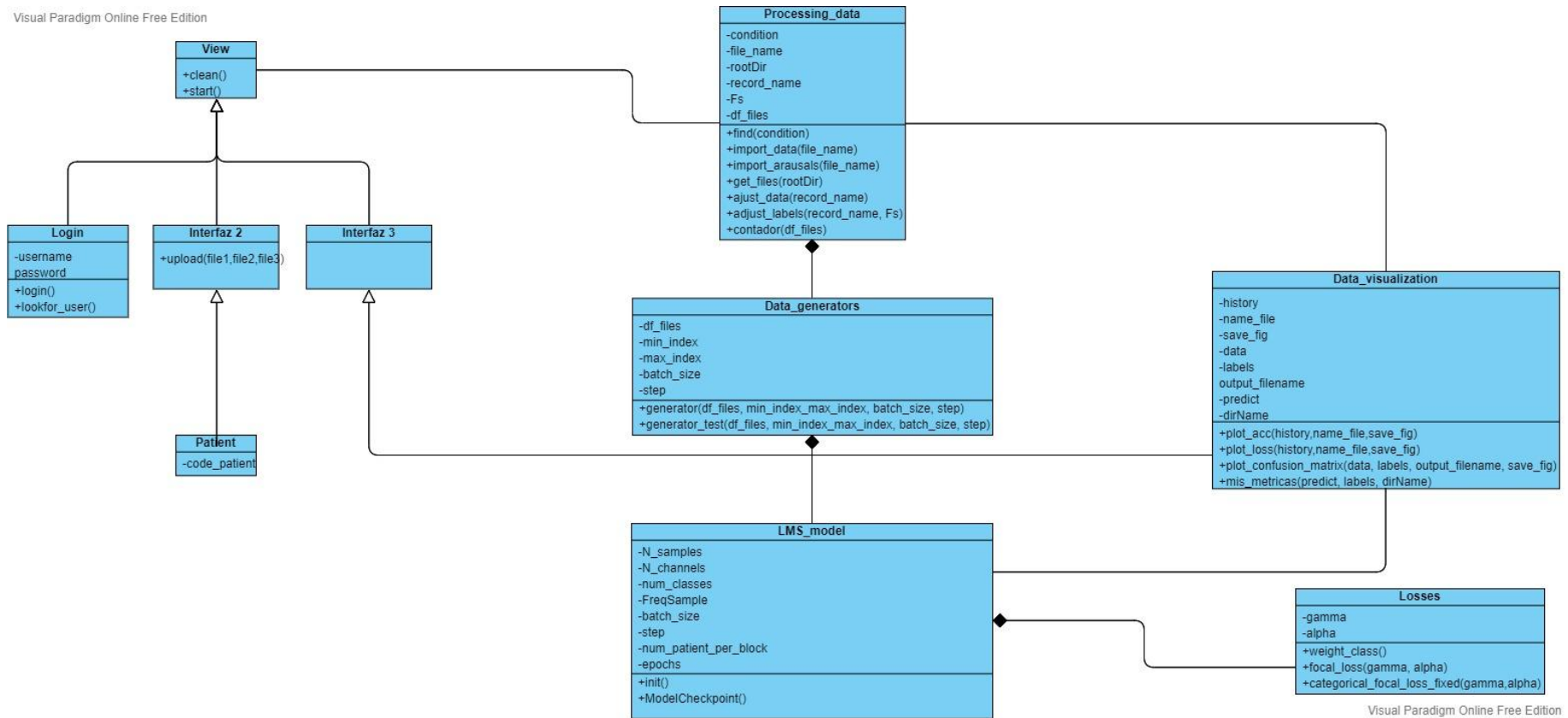
Figure 2. Classes Diagram

5.2. Uses Cases

5.2.1. Actors

5.2.1.1. User

The generic user represents a person who is medical personnel or another user that has a reason to analyze a sleep signal and has been validated through security measures such as username and password to proof of identity.

5.2.1.2. System under design

The System Under Design is a sleep analysis system that is being created (BioSleep). This actor represents the system and the actions that it takes.

5.2.2. List of Use Cases

- Login
- Analyze data
- Load files
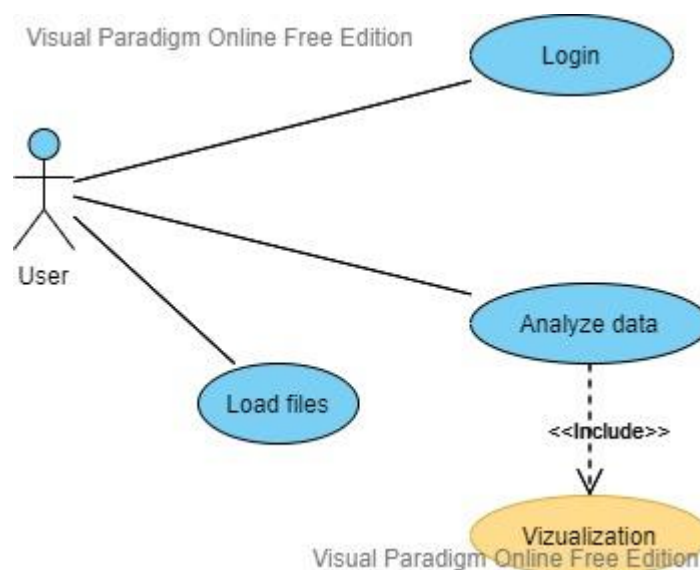- Visualization results

5.2.3. Use Case Diagrams

Figure 3. Use case diagram

5.2.4. Use Cases

CU01	Login	
Primary actor	User	
Dependencias	[FR1] Access to the system [NFR1] Access restriction	
Precondition	The user must be belonging to medical personnel. He proves his identity with his username and password.	
Flow of events	**Step**	**Action**
	1	The user set his username and password.
	2	The system verifies the data in the database.
	3	The user click on *login* button
	4	The system verifies if the user is registered in the database. The system allows the user to access the application.
Postcondition	The user access to application	
Assumptions	It is assumed that all users are registered in the databases.	
Priority	high	
State	Finalized	
Suggested Interface		

CU02	Load files	
Primary actor	User	
Dependencias	[FR3] Data loader and formats support [NFR2] Database access	
Precondition	The user must be belonging to medical personnel. He probes his identity with his username and password.	
Flow of events	**Step**	**Actions**
	1	The User selects a file/directory on the Local File List Window

	2	The User click "Upload" button.
	3	If the User uploads a file, then it will only upload the file.
	4	The system prompts the User the successful message with the number of uploaded files, directories, and the average-speed
Postcondition	The user upload files	
Priority	high	
Urgencia	iImmediately	
State	Finalized	
Suggested Interface		

CU03	Analyze data
Primary actor	User
Dependencias	**[FR3] Data loader and formats support** **[FR4] View files** **[NFR2] Database access**
Precondition	The user must be belonging to medical personnel. He proves his identity with his username and password.

Flow of events	Step	Actions
	1	The user click on *Upload* button
	2	The system checks if the folder exists
	3	The user click on *Avaliar* button
	4	The system displays a window with a progress bar while doing the data analysis

Postcondition	The user upload files
Assumptions	It is assumed that the folders and files can be accessed locally
Priority	vital
Urgencia	Immediately
State	Finalized

Suggested Interface	

CU04	Visualization data
Primary actor	System
Dependencias	**[FR6] Results viewer** **[NFR5] Display the result**
Precondition	The system has analyzed the signals.

	Step	Actions
Flow of events	1	The system shows the patient ID
	2	The system shows the EEGsignal
	3	The system shows the AUROC AUPRC, total eventos RERA and Total eventos Apnea/hipopneia

Postcondition	The user will see the results obtained
Priority	vital
Urgencia	Immediately
State	Finalized
Suggested Interface	

5.3. Data Dictionary

Class name: Processing_data		
Attributes	**Type**	**Description**
file_name/record_name	string	Path of folder of patient
Fs	int	Sampling Frequency
n_samples	int	Record size
rootDir	string	Principal path with files of Patient
df / df_files	DataFrame: string	DataFrame which contains all path files .hea, .mat and -arousal.mat
x	List of List: float	List of EEG signal segments
y	List: int	List of labels
cont	int	number of EEG segments per patient

Class name: Generators_data		
Attributes	**Type**	**Description**
df_files	string	DataFrame which contains all path files
Min_index/max_index	int	indices of the DataFrame files which are used to generate samples
batch_size	int	the number of samples per batch
step	int	the period, in timesteps, at which you sample the feature array.
num_patient_per_block	int	Number of patients to be evaluated. This case is 1.
num_classes	int	Types of respiratory events that will be identified. For this case is 3
z	int	number of segments in an EEG record

Class name: Plot_results		
Attributes	**Type**	**Description**
History	Keras object	This object keeps all loss values and other metric values in memory. The history object is

		the output of the fit operation.
output_file	string	Path to output file
save_fig	Bool	Save figure option
data	List of list: float	Confusion matrix results
labels	list	labels witch will be plotted across x and y axis
predict	float	network model predictions

Class name: Losses		
Attributes	**Type**	**Description**
y_true	tensor	Ground truth labels, shape of [batch_size, num_cls]
y_pred	tensor	Model's output, shape of [batch_size, num_cls]
Gamma	float	Default value for *focal_loss*. Focusing parameter for modulating factor (1-p)
alpha	float	Default value for *focal_loss*. The same as the weighing factor in balanced cross entropy. Alpha is used to specify the weight of different categories/labels, the size of the array needs to be consistent with the number of classes.
tr_files	DataFrame	DataFrame which contains all path files
class_weight	Dictionary	weights associated with each class to balance the dataset
FreqSample	int	Sampling Frequency

Class name: rede		
Attributes	**Type**	**Description**
y_true	list	Ground truth labels
y_pred	tensor	Model's output
te_files	DataFrame	DataFrame which contains all path files
window	int	EEG recording segment size (in seconds, AASM recommendation)

FreqSample	int	Sampling Frequency

6. Quality control

6.1. Test scripts

The test script for modules can be used as a guide using the Test_modulos.ipynb file available in the test_scripts folder. This file evaluates the criteria presented in Table 2. The GUI the interface was evaluated based on the criteria in table 3.

Table 2.

ID	Test script for modules
1	Check if **get_files** extracts and stores the file paths correctly
2	Check if **import_data** works fine
3	Check if **import_arousals** works well
4	Check that **adjust_data** actually resizes the EEG signal on fragments of [1, Fs*30, 6]
5	Check that **adjust_labels** es del tamaño esperado [n_samples/Fs*30]
6	Check that **generator_tes**t is creating the correct size batches [Batch_size, 1,Fs*30,6]
7	Check that **model_cnn1** is creating the red and loading the weight archive
8	Test if **mis_metricas** calculating the AUROC and AUPRC for the patient
9	Check that the confusion matrix is being stored in the patient folder

Table 3.

ID	Test script for GUI
1	Check if the username and password appear on the screen
2	Check if it is possible to create a new user
3	Check if when entering a user it is possible to recover the password
4	Check if clicking the exit button closes the login window
5	Check if it is possible to upload the folder with the patient's files
6	Check if the 3 files specified in [FR3] appear in the file viewer

7	Verify that clicking on the Assess button starts the analysis of the patient registers
8	Verify that the graphs of the loaded EEG signals appear
9	Verify that the graphs and results of the analysis of the EEG signals appear
10	Check if the save function is working correctly
11	Check if clicking the *"Novo usuário"* button you return to the data upload window
12	Check if clicking the exit button closes the visualization window

7. User Guide

7.1. Installation

To use the system, check the requirements.txt file that has the list of necessary libraries and then compile the BioSleep.py file in your python environment.

All files are available at: ENTREGA INF2102 PFP and GitHub https://github.com/mjaramillo16/INF2102-Entrega-Periodo-2022-1

7.2. Login

To access the system for the first time, select the *"Novo Usuário"* button (Figure 6), fill in the User and Password fields. At the end, a message should appear indicating that the registration was successful, as shown in Figure 7.
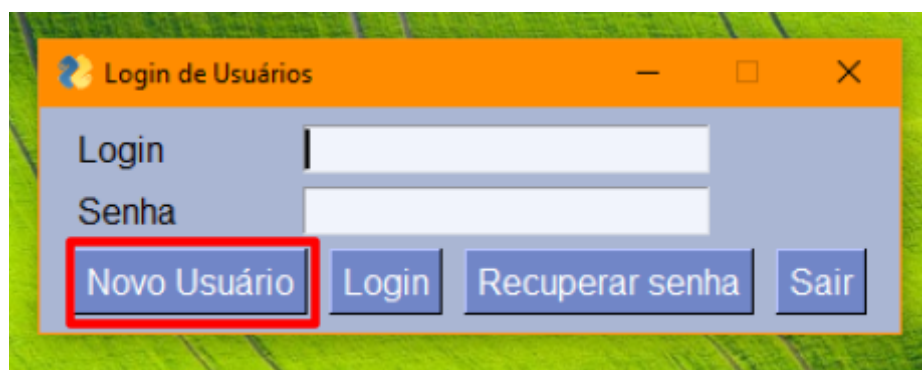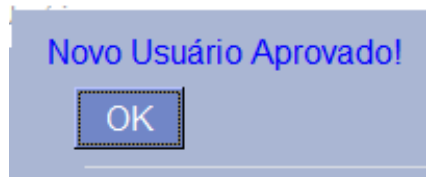

Figure 4.

Figure 5.

If you have already registered your user, fill in the User and Password fields, then select the *"login"*. At the end, a message should appear indicating that the login was successful, as shown in Figure 8.

Figure 6.

If you forgot your password, write your username and select the recover password button and the system will show the registered data.

Figure 7.

7.3. Data load

To load the patient data you want to analyze. First upload your file folder by selecting the "upload" button at the top left" (Figure 10.A). Verify that the 3 files (.hea, .mat and -arousal.mat) have been

identified by the system, and select the "analyze" button located at the bottom right of the window (Figure 10.B).
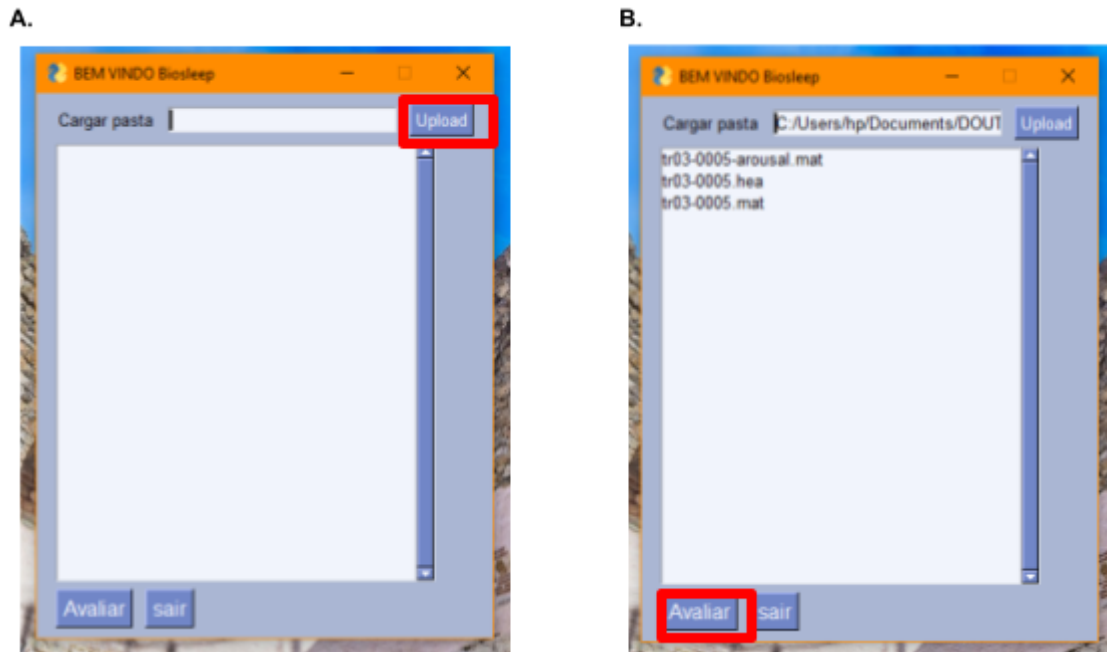

Figure 8.

Obs: The analysis of the signals will take time depending on the size of the recordings, the reference analysis time is 20 min for recordings of up to 8 hours using GPU .

7.4. Result Viewer

After finishing the evaluation of the patient's records, a new window will appear with the summary of the results as shown in Figure 12. In case of evaluating a new patient, click on the 'Novo Paciente' button. otherwise click on the 'Salvar' button and then on the 'Sair' button to close the application.
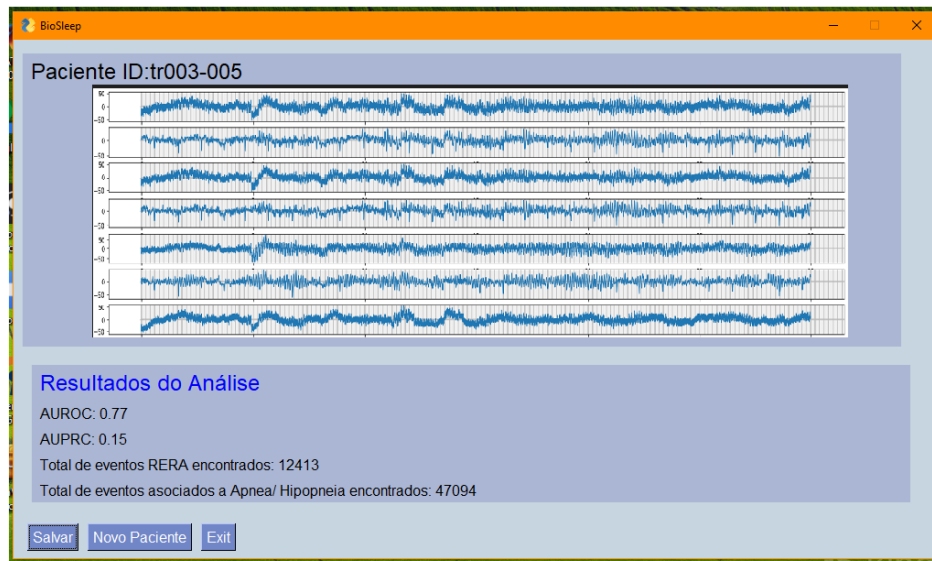
Figure 9.

7.5. Especial Instructions

Therefore of the size of the EEG records provided as an example in the data folder, to achieve a faster compilation of the system, the values of the variables z (modules\generators_data\function generator_test) and max_iter of the rede class were modified. To parse a complete record, z must be len(eeg_raw) and max_iter=None (See Figure 10).

Appendix

A1. Deep Neural Network configurations tested and results obtained

For this system, tests were performed with different numbers of convolutional layers, number of neurons and kernel sizes. The best combinations of these hyperparameters are presented in Table A1. For this network architecture, 2D convolutional layers (conv2d) were used, which allowed operating the signals in two dimensions to extract the largest amount of possible features. For each of the sampling layers (Max-pooling), kernels of size (1, 2) were used, as an optimizer Adam was used and the loss functions were switched between focal loss and binary crossentropy. The number of neurons in the FC layer is found in the #D column, and the batch sizes used are in the BS column. Finally, the TBD column expresses the data balancing technique used: i)class weight (cw) and ii) focal loss (fl).

Table A1.Deep Neural Network configurations

#	CNN*	Neurônios	Kernels	# D	BS	TBD
1	4	32,64,128,256	3,3,3,3	128	128	–
2	4	32,64,128,256	3,3,3,3	128	256	–
3	4	32,64,128,256	3,3,3,3	128	128	fl
4	4	16,32,64,128	3,3,3,3	64	128	fl
5	4	32,64,128,256	3,3,3,3	128	128	cw
6	4	16,32,64,128	3,3,3,3	64	28	w
7	3	16,32,64	7,5,3	64	128	fl
8	4	32,64,128,256	11,7,5,3	128	128	fl
9	4	16,32,64,128	11,7,5,3	64	128	cw
10	4	32,64,128,256	7,5,3,3	28	28	fl

*CNN:Number of CNN layers; # D:Neurônios Denselayers; BS: Batch size; TBD: Technique of dice rolling

Table A2.Deep Neural Network results obtained

#	Accuracy			AUROC*	AUPRC*	Normal*	RERA*	ApHip*	EBC**
	Treino	Val	Test						
1	0,9537	0,7573	0,8430	0,8940	0,8165	118505 (95,53 %)	8139 (65,57 %)	28089 (60,26 %)	154733
2	0,9269	0,7955	0,8378	0,8849	0,8048	116525 (93,94 %)	3366 (27,12 %)	33888 (72,70 %)	153779
3	0,8010	0,7488	0,7684	0,8823	0,7838	11373 (91,72 %)	6210 (50,03 %)	21060 (45,18 %)	141043
4	0,7704	0,7459	0,7530	0,8618	0,7287	118421 (95,47%)	2780 (22,40 %)	17005 (36,48 %)	138.206
5	0,8959	0,7800	0,8177	0,8676	0,7904	123559 (99,61 %)	4134 (33,31 %)	22391 (48,04 %)	150.084
6	0,8643	0,8148	0,8404	0,8885	0,8141	116327(93,78 %)	8689 (70,00 %)	29250 (62,75 %)	154266
7	0,8162	0,7146	0,7824	0,8855	0,7803	108742 (87,66 %)	5407 (43,56 %)	29471 (63,22 %)	143620
8	0,7842	0,7032	0,7659	0,8703	0,7516	108178 (87,21 %)	2005 (16,16%)	30396 (65,21 %)	140579
9	0,8721	0,8186	0,7558	0,8931	0,8199	108059 (87,11 %)	2279 (18,36 %)	28387 (60,90 %)	138725
10	0,8266	0,7261	0,8170	0,8926	0,7862	109113 (87,96 %)	5776 (46,53 %)	35074 (75,25 %)	149963
				TEC**		124045 (100 %)	12413 (100 %)	46613 (100 %)	183552

∗ Calculated for the Test set***ApHip: apneia/hypopneia; TEC:Total Examples by Class; EBC: Total Examples Classified

In the Table A2, it can be seen that the best result was obtained for configuration #1, reason why the weight file associated with it is used for this project (cnn1_t1.hdf5)

A2 . Test Cases

Data: 22/08/22
Responsible: Maria Leandra
ID_TestCases: TC012

ID	Test script for GUI	OK
1	Check if the username and password appear on the screen	yes
2	Check if it is possible to create a new user	yes
3	Check if when entering a user it is possible to recover the password	No
4	Check if clicking the exit button closes the login window	No
5	Check if it is possible to upload the folder with the patient's files	Yes
6	Check if the 3 files specified in [FR3] appear in the file viewer	No
7	Verify that clicking on the Assess button starts the analysis of the patient registers	Yes
8	Verify that the graphs of the loaded EEG signals appear	No
9	Verify that the graphs and results of the analysis of the EEG signals appear	No
10	Check if the save function is working correctly	No
11	Check if clicking the "*Novo usuário*" button you return to the data upload window	yes
12	Check if clicking the exit button closes the visualization window	yes
ACTIONS: Verify actions in while loop of the BioSleep.py		

Data: 24/08/22
Responsible: Maria Leandra
ID_TestCases: TC020

ID	Test script for GUI	OK

1	Check if the username and password appear on the screen	yes
2	Check if it is possible to create a new user	yes
3	Check if when entering a user it is possible to recover the password	yes
4	Check if clicking the exit button closes the login window	yes
5	Check if it is possible to upload the folder with the patient's files	Yes
6	Check if the 3 files specified in [FR3] appear in the file viewer	yes
7	Verify that clicking on the Assess button starts the analysis of the patient registers	Yes
8	Verify that the graphs of the loaded EEG signals appear	No
9	Verify that the graphs and results of the analysis of the EEG signals appear	No
10	Check if the save function is working correctly	No
11	Check if clicking the *"Novo usuário"* button you return to the data upload window	yes
12	Check if clicking the exit button closes the visualization window	yes
ACTIONS: Verify actions in while loop of the BioSleep.py		

Data: 29/08/22 **Responsible: Maria Leandra** **ID_TestCases: TC020**		
ID	**Test script for GUI**	**OK**
1	Check if the username and password appear on the screen	yes
2	Check if it is possible to create a new user	yes
3	Check if when entering a user it is possible to recover the password	yes
4	Check if clicking the exit button closes the login window	yes
5	Check if it is possible to upload the folder with the patient's files	Yes
6	Check if the 3 files specified in [FR3] appear in the file viewer	yes
7	Verify that clicking on the Assess button starts the analysis of the patient registers	Yes
8	Verify that the graphs of the loaded EEG signals appear	yes

9	Verify that the graphs and results of the analysis of the EEG signals appear	yes
10	Check if the save function is working correctly	No
11	Check if clicking the "*Novo usuário*" button you return to the data upload window	yes
12	Check if clicking the exit button closes the visualization window	yes
ACTIONS: Verify actions in while loop of the BioSleep.py		