Identifying Patterns in Jaw Activities: Time Series Analysis of Sleep-Related Data

Clément Detry
Department of Data Science and Artificial Intelligence
Maastricht University
Maastricht, The Netherlands

***Abstract*—This study constitutes a first approach for the implementation of deep learning models dedicated to the classification of JAWAC[1] signal validity. Deep learning methods such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been shown to provide state-of-the-art results on the classification of time series data with little or no engineering of data features [2], [3]. Two different types of models are proposed in this paper, a binary one aiming at detecting valid and invalid signal areas and another multinomial model capable of detecting a third new awake class. The final product is designed to be integrated into Nomics Sleep & Care platform to help doctors track the signal quality of their patients. This study applies and compares both the CNN and RNN models using a dataset of approximately 22,000 patient analyses from 2019. The best results are obtained by using a balanced dataset, implementing the SMOTE oversampling technique of minority classes. The proposed binary network achieved a testing accuracy of approximately 99.8%, while the multi-class model achieved about 98.5%.**

***Index Terms*—deep learning, time series classification, convolutional neural network, long-short term memory, sleep apnea**

I. Introduction

Analysis and classification of clinical time series data in physiology and disease processes are considered catalysts for biomedical research and education. Innovative computational tools for the classification of physiological data are increasingly needed to facilitate research on new and challenging unsolved clinical and life sciences problems [4]. Nowadays, there are many sleep-related diseases recognized worldwide. The illness this research paper focuses on is called sleep apnea and is being investigated by Nomics, the Belgian company cooperating in this study. This pathological condition is very serious and affects more than 4% of the population. In 2018, an estimated 22 million Americans suffered from moderate to severe sleep apnea [5]. It is essential to diagnose it because this disorder can be related to obesity, cardiovascular risk or even road accidents and must be monitored and treated. The clinical tests to diagnose this type of disease are expensive and inconvenient for the patients due to the environment and the circumstances of the examinations. In these traditional tests, patients must spend the night in the hospital under surveillance with a multitude of sensors and wires attached, which sometimes skews the sleep results. Therefore, Nomics has been working for more than 15 years on developing portable electronic devices capable of determining the breathing effort during sleep using only two sensors. One on the chin and one on the forehead, used to capture the human mouth opening. This signal is called JAWAC. In this configuration, patients can test themselves at home, in their familiar surroundings. Hence, the results are more consistent and less biased.

A. Problem Statement

One of the main problems in this testing setup is the lack of direct feedback during the patient examinations. Sometimes, this could lead to invalid analyses. For example, when a sensor is disconnected or moved for too long overnight. In the worst cases, patients must be retested if the total diagnosis time is outside certain conditions. Unfortunately, each uploaded graph must be manually reviewed by a Nomics operator, which slows down the process. Hence, the objective is to have an algorithm that will automatically process the signal beforehand to be able to tell the doctor if their patient has been correctly examined or not in real time.

First, the task is to deal with a two-class classification problem, whose objective is to provide a learning model capable of identifying valid and invalid signal regions. According to the obtained result, a decision must be taken on the overall quality of the signal, following this simplified rule. If there is a continuous 4-hour block or at least two continuous 2-hour blocks of validity, the entire analysis will be retained for diagnosis. Next, a third class of interest is introduced to detect areas in the signal where the patient is not sleeping. This new type of signal is considered valid since the sensors are measuring correctly. Still, it must be removed from the total amount of signal diagnosis time to have a more pertinent statement for the decision making of the entire analysis.

B. Related Work

There are many methods developed for time series classification in different fields of applications. Most of them fall into two major groups.

The first cluster includes classification techniques that use algorithms based on morphological features. Among them, the most famous are distance-based like k-Nearest

This thesis was prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Data Science and Artificial Intelligence, Maastricht University. Supervisor(s): Rachel Cavill, Didier Leclercq (CEO at Nomics) and Pierre Ansay (Engineer at Nomics)

[1]JAWAC is the name assigned to a specific type of signal that measures human breathing effort during sleep through the opening of the jaw, created by Nomics Sleep & Care[1]

Neighbor (KNN), interval-based such as Time Series Forest or even dictionary-based like Bag-of-SFA-Symbols (BOSS) [6]. For the distance-based approach, clustering is conducted by comparing the likeness between two time series using a distance metric. Next, interval-based procedures choose one or more segments of the series and use summary statistics as features for classification. Finally, dictionary-based methods categorize time series based on the frequency of their recurring sub-series [4]. However, one thing unifies these methods, they all require some form of feature engineering as a separate step before proceeding to classification [7].

The second group includes more recent approaches to classifying time series data that use deep learning. Deep learning models are types of neural networks consisting of several layers of neurons. They generally have many more parameters than other machine learning models. Often, they are made up of a core of layers to which the data is transmitted. While the first layers describe the basic patterns, the last layers encode semantically significant representations. Thus, the output layer of the system has to contain what is relevant and prominent in the data. Over the past decade, deep learning models have become the overwhelming choice for image and language classification. It has only recently grown in popularity for time series, and they now have outperformed many basic learning techniques [8], [4]. These models are called recurrent (RNNs) and convolutional neural networks (CNNs). These techniques have been shown to provide state-of-the-art results on complex activity recognition tasks with little or no data feature engineering [2].

The most widely used RNN architecture for this particular task is the Long-Short Term Memory (LSTM), because they can catch long-term time dependencies, they have been successfully employed to solve many demanding bioinformatics and computational biology problems [4]. As a state-of-the-art technique for learning physiological models, many uses of LSTM and other deep learning networks have recently been recommended in the literature, such as the classification of motor imagery, mental load, seizures or even sleep stage [4].

C. Research Goals

This work aims to build a classification algorithm to replace the human task of manually checking for signal validity after each uploaded analysis and thus speed up the entire process. To accomplish this, a large number of consistent labeled data are required to train any model. Therefore, Nomics has made available all data recorded during 2019, the last full year before the worldwide health crisis. First, research is needed to understand the data, how to process and engineer it to make it suitable and efficient for specific training purposes. The next step is to determine which machine learning models are best suited to the problem statement. Finally, using specific methods of adjustments of the models' parameters to optimize their efficiencies.

In a nutshell, the research questions explored in this study are:

1) How to design a dataset with human physiological time series data suitable for validity classification?
2) How does data balancing impact the classification behaviour in the time series framework?
3) Are CNNs and RNNs best suited to extract features in time series data to perform binary validity classification?
4) How do the model's performances change after increasing the complexity of the problem by adding a third awake class in addition to the previous valid and invalid ones?
5) Do hyperparameter fitting techniques provide better models for higher accuracy and global robustness performance?

II. Methodologies

A. Data Collection

The first challenging task of this study was to gather data from the company. Due to the large amount of patients analysis files to be transmitted, about 22,000 in total, this represents over 40 GB of data. After collecting them all, it was necessary to decode the files stored in a compressed edf format. This type of file extension is a simple and flexible format for exchange and storage of multi-channel biological and physical signals [9]. There exists an open source python library maintained by a community of scientists and research labs that can perform this task, called mne [10]. Converting the edf files one by one into usable python data took between 5 and 30 seconds each, depending on the duration of the analysis.

B. Data Preprocessing

Then, the dataset needs to be preprocessed before being manipulated by learning models. The better the data is structured, the higher the performance of the classifiers. Therefore, data needs to be cleaned, and missing values has to be dropped [11]. For this purpose, the pandas library is used. Pandas is intended to be the high-level building block for practical, real-world data analysis in Python [12]. Its utility will be crucial when it comes to manipulating data in smart and fast ways.

The first preprocessing task consists in reducing the resolution of the time series. Nomics devices record data on the breathing effort of patients at a frequency of 10Hz. Hence, for a total recording time of 8 hours, the time series will contain 288'000 data points. This is a massive amount of information for a single analysis. The resolution may be too high for the complexity of the task. Consequently, various types of datasets are built with different downsampling rates for testing. For instance, to reduce the resolution of the series from 10Hz to 1Hz, the median point for each second is taken. The median value is preferred to the mean in this context because the data may be very sensitive. In order to keep a meaningful representation of the signal without losing too much information, the most robust technique against outliers is recommended [13].

The second data manipulation involves transforming input vectors to have samples with the same length. As shown

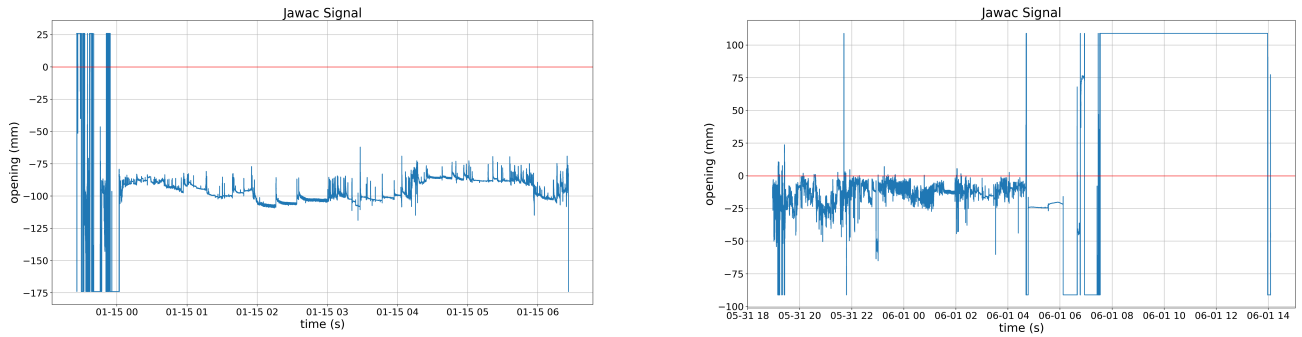

Fig. 1: Examples of JAWAC signals

in Figure 1, different signals do not always have the same length of recording time. To overcome this issue, there are several methods, among which the two best known are padding and segmentation. The first consists of adding zeros either at the beginning or at the end of the time series until the maximal sample size of the dataset is reached. With this configuration, all input data would have the same length. Each time series is linked either to a valid or invalid label, depending on the complete sequence of data. In contrast, the segmentation technique divides each time series into small windows of the same given size. Hence, the signal is divided into fragments with their respective labels, providing information about the nature of the tiny region. To classify a new patient analysis, the same process is applied. The time series is divided into small portions where each of them receives a predicted label. After processing the whole new sequence, each window is combined to see how many valid signal patterns are found in total. This second segmentation technique is the most appropriate to tackle this particular problem, as a flat signal is related to invalidity and using padding by joining consecutive sets of zeros would lead to the misclassification of entire series.

Then, another essential point to keep an eye on is the balancing of the data. Imbalanced data refers to the types of datasets in which the classes have an unequal distribution of observations. This means that one class label has a very high number of instances while the other has very few. The difficulty of dealing with unbalanced datasets resides in the fact that most machine learning techniques overlook the minority class, resulting in poor performance on that cluster, even though it is usually the behaviour of the minority class that is most critical. One approach to dealing with imbalanced sets of data is to upsample the smaller class. The easiest way is to randomly duplicate instances in the minority group. However, these new examples do not contribute any additional knowledge to the model. Instead, new instances can be replicated from the existing ones. This is a form of data augmentation of the smaller class. The method used in this study is called the Synthetic Minority Oversampling Technique (SMOTE) [14]. It focuses on the feature space to produce new samples by first picking an instance of the minority group. Next, the k-nearest neighbors from that class are retrieved. One of them is used to interpolate a new synthetic example. To do so, the difference between the feature vector and its selected neighbor is calculated using a distance metric technique. The new instance is finally multiplied by a random value between 0 and 1 before being added to the feature space [15].

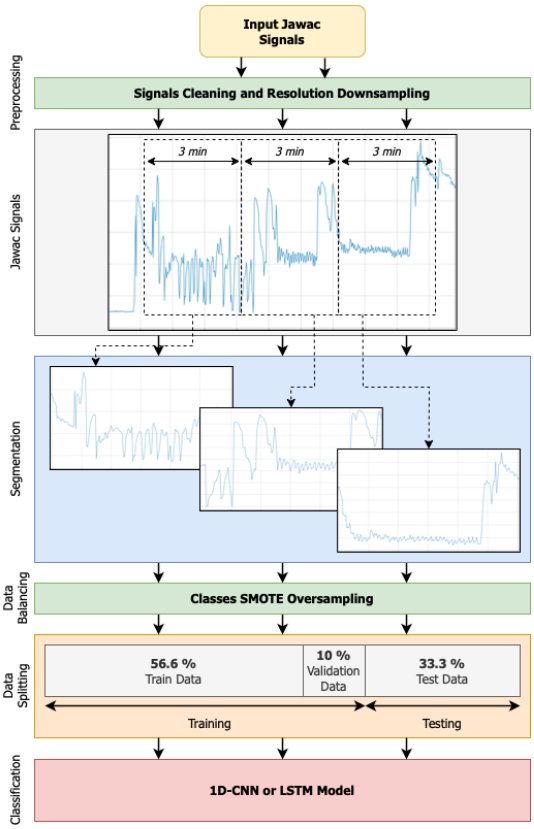

Fig. 2: Data processing recap

To clarify the data process and make it easier to understand, Figure 2 shows a complete summary of each step. After the SMOTE oversampling process is finished, the data can be seen to be divided into multiple sets. This corresponds to the data splitting step. They all have a specific role in the training of the models. The most significant part is reserved for the training of the neural network. It constitutes 66% of the total input data. Nevertheless, 10% of this subset is reserved for the validation cluster used to test the neural network after each epoch to ensure that it does not overfit

the training data. The remaining 33% is reserved for testing the model after the training process is done to estimate its reliability on unknown data.

C. Convolutional Neural Networks (CNN)

CNN is the first deep learning approach applied to the data in this paper. Consider a temporal series of length n and width k. The length is the number of time steps, and the width is the number of tags in a multivariate time series. In the scope of this work, only one JAWAC signal channel is used at a time, which means k is equal to one. In order to extract deep features from the temporal sequences, the four steps mentioned in figure 3 are applied.

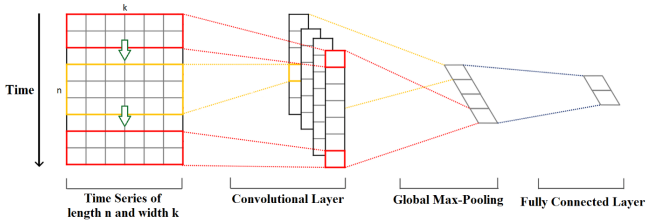

Fig. 3: 1-D Convolution for Time Series [16]

Baseline CNN Architecture:

The basic architecture adopted to perform the classification task is shown in Figure 4. It starts with a 1-dimensional input convolution layer that produces 16 output filters using 2x2 kernels. The dimension of the output filters is further reduced with a 2x2 max-pooling layer to keep the most relevant part within each filter vector. In addition, a flattening layer is required to convert everything into a single array. This vector is finally loaded into a simple dense layer with the number of classes as an output size. The activation function can be either sigmoid or softmax, depending on the type of classification, binary or multinomial.

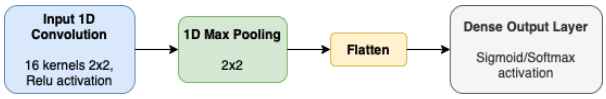

Fig. 4: CNN baseline architecture

D. Recurrent Neural Network (RNN)

RNN is the second deep learning approach used for this study, and more precisely, the Long-Short Term Memory variant. LSTM is a deep recurrent neural network architecture used for the classification of time series data. Time frequency and space properties of time series are presented here as a robust framework for LSTM treatment of long sequential data [4]. It is distinguished from a regular feed-forward network because its architecture includes a feedback loop as it can be seen in Figure 5. A particular unit, called a memory cell, retains past information for a more extended period of time to make an accurate prediction [17].

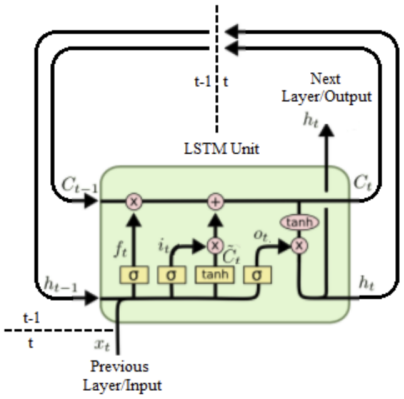

Fig. 5: LSTM cell structure [18]

Baseline LSTM Architecture:

For this model, the proposed basic architecture contains only two elements. The first one is the LSTM layer. For this level, 10 units are set up, which represent the dimensionality of the outer space. Therefore, it means the layer contains 10 independent copies of the LSTM cell shown in Figure 5. They have the same structure but since they are initialized with different sets of weights, they compute different patterns. The second element is a fully connected layer used to predict the time series instance. A visual representation of the architecture is given in Figure 6.

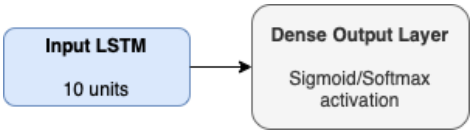

Fig. 6: LSTM baseline architecture

E. Hyperparameter Tuning

Many model parameters can be adjusted to better fit the problem task, whether by the number of nodes, hidden layers, activation functions, number of epochs or many others [19]. There are no fixed rules to conform to precisely define the perfect architecture to use. However, intuition and a little trial and error are the best solutions to help achieve better general outcomes [20]. An experiment is designed to determine whether more complex models could be more efficient to solve the time series classification problem and under which conditions.

III. Experiments

Regarding the following experiments, they all share the same dataset to obtain the most consistent results possible. A small subset of the 22,000 proposed analyses were retained. Most of the sequences were not accurately labeled in the files sent by the company, causing too much bias in the training of the models. Therefore, it was necessary to graphically verify each signal to extract the neural network's maximum potential. For this reason, only 300 of them were kept. Since

the preferred solution was to segment the temporal data into windows of a couple of fixed minutes, the number of samples to train the models was still high enough. The reduction of the dataset also allowed for a faster computation in order to have enough time to run all the tests.

The experiments are organised into 5 sections, each designed to address the research objectives established, respectively labeled from A. to E. The first one explores the dataset's creation through the whole data processing phase to see how many samples can be obtained with the different segmentation and downsampling configurations, along with the computational speed required. The second focuses on data balancing by experimenting with the learning models on different balanced sets. Third one compares the two basic CNN and LSTM models to determine which one is more successful in predicting the valid and invalid small signal windows. The fourth experiment is similar to the previous one, except the addition of the third new awake class. The final one consists of optimizing the hyperparameters of the different models to obtain higher prediction scores.

Remember that for each model training, the dataset is divided into two different subsets, the training set, composed of 66.6% of the data, and the test set, consisting of the rest 33.3%. Additionally, a validation set of 10% total size of the data set is used from the training set.

A. Framework

The experiments were all carried out using TensorFlow 2.6 with Python 3.8. TensorFlow is an end-to-end open-source platform for many various tasks in machine learning [21]. On top of that, the Keras deep learning API library was applied [22]. For the setting of the hyperparameters, the official method of keras tuner was used.

Concerning the hardware, the experiments were conducted on a laptop composed of a 6-Core Intel i7 with an Intel UHD graphics 630 GPU and 16GB Ram.

B. Metrics

Once a classification model is built, it is necessary to evaluate the quality of the predictions made by the classifier. Using confusion matrices is an excellent way to measure the efficiency of networks. Some interesting values can be extracted from them, such as the accuracy, the precision, the recall or the F_1-score. For this specific study, the models are essentially evaluated on the following two complementary metrics [23].

1) The first one is accuracy. It is used to inform on how many instances the test set was correctly classified, telling how accurate the model is (equation[2] below [23]).

$$\text{accuracy} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}}$$

2) The second one is the F_1-score. This metric is more relevant to the number of wrong decisions made by the models (equations below [23]).

$$\text{precision} = \frac{\text{TP}}{\text{TP + FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP + FN}}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision + recall}}$$

In addition to these measurements, the results predicted by the model can be evaluated using confidence intervals. These are a way to quantify the uncertainty of an estimate. Thus, the probability of classification under a certain level of confidence can be estimated [24]. For the following experiments, a 95% confidence interval estimate of the accuracy is considered.

Furthermore, the computational time of model learning is also measured as it plays an important role. Since future data collected by Nomics will be retained to continue to train and improve the performance of the models, their time efficiency ratio is essential to consider.

Lastly, other data are evaluated for the different classifiers. Learning curves, including training and validation accuracies, F_1-scores, and losses over epochs, can provide insight into how well the models learn over time. They can also reveal overfitting or underfitting of the training data.

For this specific research, models are considered to perform well if they have a high accuracy, a high F_1-score, and consistent learning curves. Accuracies and F_1-scores are deemed high at 90% and above. The best model settings are shown in bold in the results tables. The criteria to compare and choose the best configurations are the prediction efficiency according to the result metrics and the training speed.

IV. Experiments Results

A. Dataset Creation

Table I provides information on the preprocessing of the data set. It can be observed that the number of samples decreases drastically and inversely to the window size. The total computation time is also more impacted by a large number of samples than a large sample size. This table will be used as a reference to determine the best models when the performances of several classifiers are very close.

B. Data Balancing

Table II refers to the data balancing for both binary and multinomial datasets. On the left side of the table, two classes are represented almost equally, 56.1% of valid instances and 43.9% of invalid ones. However, there is a

[2]TP, TN, FP and FN repectively corresponding to the true positive, true negative, false positive and false negative classified instances.

[3]This column corresponds the total number of samples in the processed dataset for the different size of window segmentation settings.

[4]Sample size value is related to the total length of each small time series instance in the processed dataset (= window size (sec) * resolution (Hz)).

TABLE I: Dataset Preprocessing Info

window size	#samples[3]	resolution (Hz)	sample size[4]	computation time (sec)
1 minute	127666	10	600	493.17
		5	300	402.7
		1	60	409.22
		0.33	20	362.7
		0.2	12	302.25
2 minutes	63729	10	1200	216.76
		5	600	177.69
		1	120	134.34
		0.33	40	128.04
		0.2	24	124.87
3 minutes	42431	10	1800	167.34
		5	900	128.06
		1	180	85.1
		0.33	60	80.76
		0.2	36	79.04

considerable gap between the three classes on the right side, with the awake group being much less represented. The hypothesis would be that the binary classification might not be too affected by the slight imbalance. However, the three-class classification may encounter problems with the awake cluster due to its significant minority. The accuracy of the classifier could decrease due to the poor performance on this class.

TABLE II: binary and Multinomial data balancing

Binary		Multinomial		
%invalid	%valid	%invalid	%valid	%awake
56.1	43.9	56.0	36.1	7.9

After running 10 trials of both CNN and LSTM baseline models with balanced and unbalanced data, the average results are plotted in Figure 7. It can be seen from the left graph that the accuracy is not affected at all by the small data imbalances, as expected. On the other side, there is a clear difference (about 12.5% accuracy) between the learning curves of the model using both types of data.

C. Binary Baseline Models: CNN vs LSTM

Table III summarizes the prediction results of the baseline binary models on different possible configurations of the dataset after 5 epochs. The 5Hz and 0.33Hz time series were removed from this experiment since they did not add any new information to the overall results. In general, both types of models performed very well on this binary classification with an average accuracy and F_1-score above 99% for each different segmentation type and downsampling resolution. The best results obtained for the CNN and the LSTM classifiers are highlighted in bold in Table III. With a slight preference for the LSTM model trained on 3-minute temporal data samples with a resolution of 1Hz because of its high prediction results.

TABLE III: 5 Epochs Binary Baseline Models Recap

	window size	resolution (Hz)	train acc	train F_1	val acc	val F_1	train epoch time avg. (sec)
CNN	1 minute	10	99.58	99.55	99.67	99.61	7.81
		1	99.64	99.63	99.62	99.59	2.75
		0.2	96.21	95.36	96.1	95.75	2.12
	2 minutes	10	99.59	99.56	99.63	99.6	6.05
		1	**99.63**	**99.6**	**99.51**	**99.47**	**1.88**
		0.2	94.13	92.43	95.72	94.69	1.17
	3 minutes	10	99.63	99.75	99.69	99.7	6.71
		1	99.7	99.67	98.9	98.95	1.53
		0.2	93.75	93.12	92.89	92.65	0.89
LSTM	1 minute	10	99.72	99.72	99.67	99.67	272.72
		1	99.76	99.75	99.8	99.8	31.98
		0.2	99.36	99.36	99.44	99.45	8.86
	2 minutes	10	99.73	99.73	99.77	99.77	262.84
		1	99.74	99.74	99.78	99.79	28.78
		0.2	99.56	99.55	99.68	99.68	6.7
	3 minutes	10	99.69	99.69	99.78	99.76	258.88
		1	**99.8**	**99.8**	**99.84**	**99.84**	**27.77**
		0.2	99.48	99.47	99.31	99.33	6.27

Figure 8 provides a more in-depth look at the results of the best model. It can be noticed that the baseline LSTM model learns very quickly to classify valid and invalid regions. After one epoch, the validation and training accuracy curves are already above 97%. This observation makes the training speed column of the models in Table III less important, given their learning curve. The classifier does not overfit the data either, as the validation scores remain as high as the learning curves over the epochs. From the confusion matrix of tested and predicted instances, a 95% confidence interval indicates that the true classification accuracy of this model is between 99.62% and 99.79%.

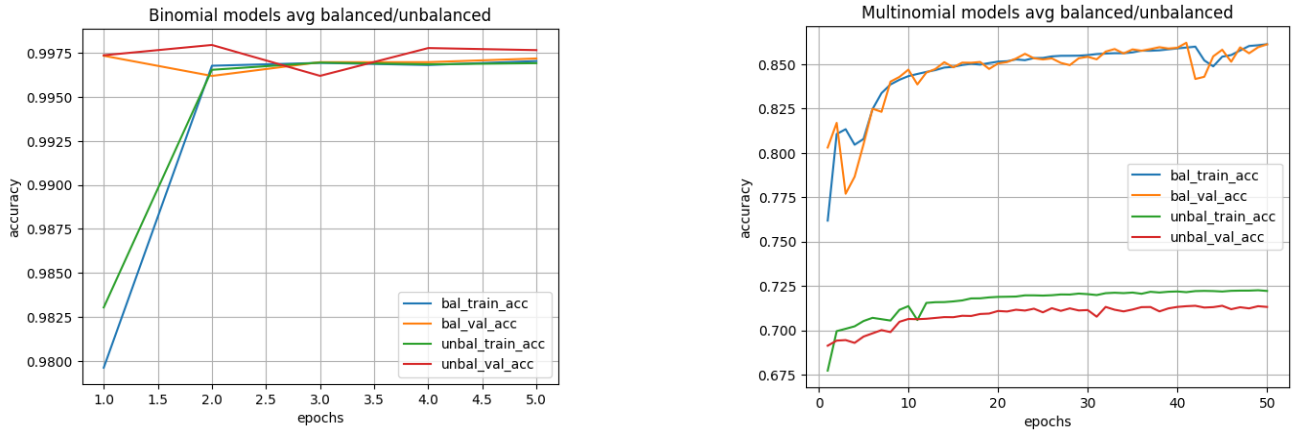

Fig. 7: Balanced vs. unbalanced dataset models accuracies

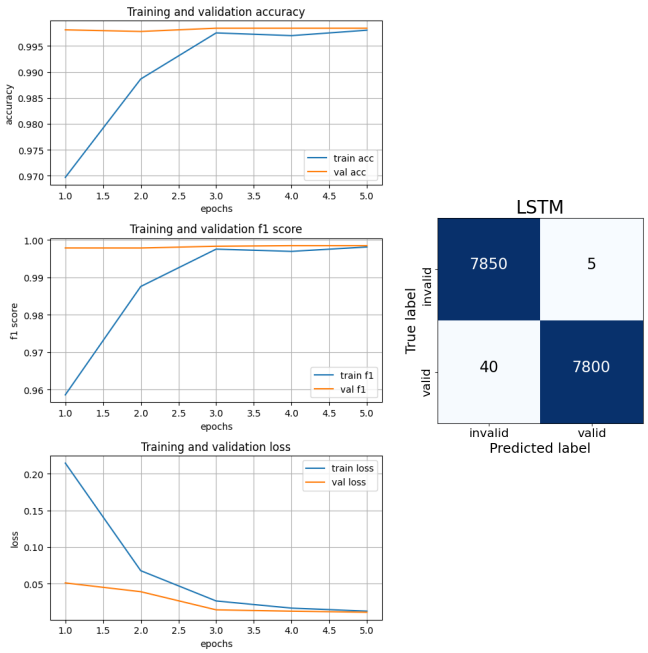

Fig. 8: Baseline binary LSTM model

D. Multinomial Baseline Models: CNN vs LSTM

Table IV is somewhat similar to the previous Table III. However, it sums up the performance of the models on the three-class problem on 50 epochs. It can be directly observed that the results are slightly less accurate than for the two-class problem with the new awake cluster. The LSTM model struggles to obtain good results above 90% and is less efficient on test sequences with a high resolution of 10Hz than on downsampled ones of 1Hz. Conversely, the CNN trains better on longer 3-minute signals with a frequency of 10Hz. Reaching a training accuracy and F_1-score of approximately 96.4% and validation results around 95.2% after 50 epochs. Moreover, the average training time of the models for each epoch is mainly in favor of the CNN. The calculation times seem to increase linearly for the CNN, while for the LSTM, each increment of the sample resolution size exponentially affects its speed performances.

TABLE IV: 50 Epochs Multinomial Baseline Models Recap

	window size	resolution (Hz)	train acc	train F_1	val acc	val F_1	train epoch time avg. (sec)
CNN	1 minute	10	93.47	93.47	94.13	94.13	14.57
		5	89.53	89.52	89.51	89.51	6.72
		1	83.47	83.42	83.37	83.35	3.74
	2 minutes	10	94.78	94.79	93.74	93.73	8.51
		5	92.07	92.07	92.1	92.09	7.27
		1	85.04	85.03	82.99	83	2.52
	3 minutes	**10**	**96.41**	**96.42**	**95.23**	**95.13**	**11.35**
		5	93.61	93.63	92.53	92.53	3.95
		1	86.62	86.61	87.11	87.07	2.07
LSTM	1 minute	10	78.46	78.46	78.24	78.19	402.93
		5	85.4	85.4	83.98	83.97	202.15
		1	88.77	88.77	88.86	88.94	43.14
	2 minutes	10	79.9	79.89	80.39	80.44	458.41
		5	80.85	80.83	81.62	81.57	203.25
		1	**90.63**	**90.64**	**90.49**	**90.46**	**42.01**
	3 minutes	10	73.64	73.34	75.37	74.9	405.92
		5	82	81.98	82.84	82.82	204.46
		1	85.48	85.45	84.81	84.84	43.27

Once again, a closer look at the results plotted over the epochs of both models best performance is presented in Figure 9. The 3 leftmost graphs represent the results obtained with the CNN using the boldly highlighted row dataset configuration in Table IV, and the 3 rightmost graphs represent the same for the LSTM. The learning speed is slightly slower than for the previous binary problem. The prediction performance seems to keep increasing on both sides. Nevertheless, it can be noticed that the CNN slightly overfits the training data considering the validation curves for the accuracy and the F_1-score. This is even more visible on the lower left graph, where the validation losses keep increasing starting from the 10th epoch. This problem may be resolved by tuning the hyperparameters of the model. The LSTM seems to be relatively stable in its learning, even if its prediction results are slightly lower than those of the CNN after 50 epochs.

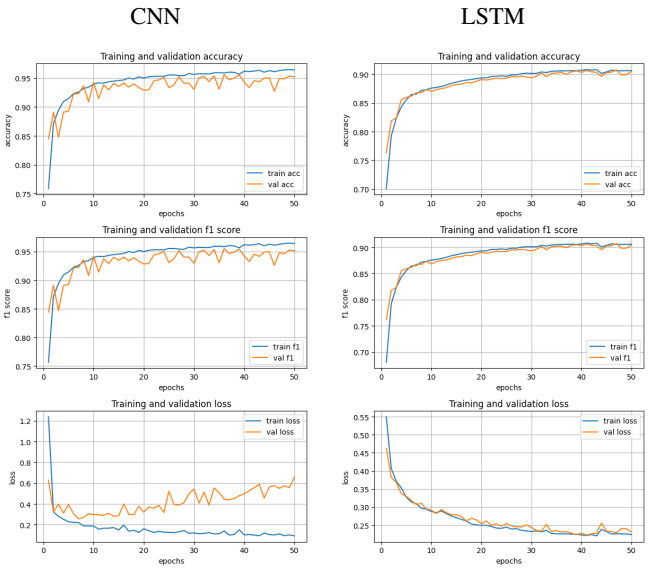

Fig. 9: Baseline multinomial models metrics plots

Looking at the confusion matrices in Figure 10, representing the CNN and LSTM results previously described in Figure 9, several interesting insights can be derived.

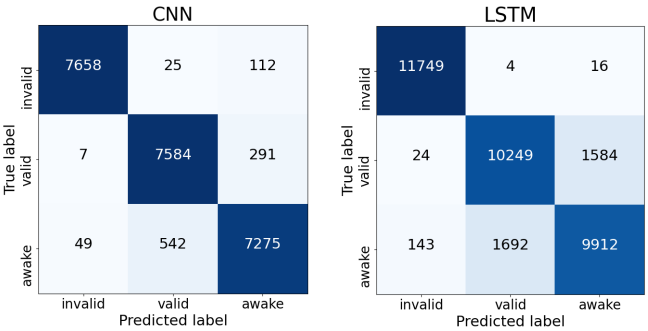

Fig. 10: Baseline multinomial models confusion matrices

With a 95% confidence interval, the valid classification is likely between 95.3% and 95.9% for the CNN model, and between 89.9% and 90.5% for the LSTM.

One point that can be noticed with these confusion matrices is that the correct classification of invalid samples has a large influence on the accuracy and F_1 performance results. Indeed, more than 98.2% of the invalid samples are correctly classified for both matrices and less than 1.3% are mispredicted. While for the right matrix illustration, 82.9% of the awakening instances are correctly classified and 13.8% are wrongly predicted.

Apart from that, the CNN seems to dissociate the portions of valid and awake signals better than the LSTM. However, the balance between valid instances classified as awake and conversely is more stable on the LSTM side.

One last small remark that can be mentioned is that the LSTM model will be slightly more likely to skip invalid instances than the CNN, but will be a little less susceptible of predicting incorrect samples as invalid.

E. Hyperparameter Tuning

After running the keras tuner on 100 random parameters configurations of the two models, these two following architectures are the most efficient found.

New CNN architecture:

Figure 11 is the updated version of the CNN model in Figure 4. Compared to its previous baseline architecture, the revised one has a 1-dimensional hidden convolution layer. The first two layers now produce 64 filter vectors, increasing the outer space. A dropout function of 0.5 is added before the max-pooling layer. It is used to prevent the model from being overfitted by randomly setting the outgoing edges of hidden neurons to 0 with a frequency of 0.5 at each update of the learning phase [25]. The last updated component is a new 100-unit hidden layer inside the fully connected network.

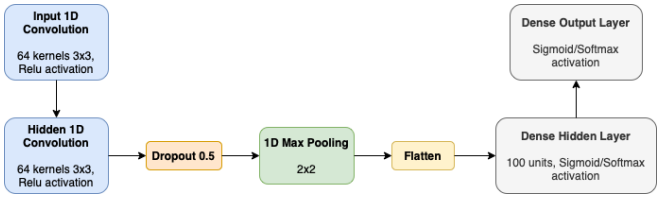

Fig. 11: CNN tuned architecture

New LSTM architecture:

Regarding the LSTM model, its hyperparameters also changed after the optimizer was run. Figure 12 shows two new components in the architecture compared to Figure 6. Similarly to the tuned CNN model, a dropout function is included to avoid overfitting, and the identical hidden layer of 100-units. This approach is more robust to tackle more complex problems.

Table V displays the new prediction results obtained with the tuned networks over 50 epochs. Only the multi-class problem on the best dataset configurations was tested, as

Fig. 12: LSTM tuned architecture

the binary problem already performed well with the previous baseline models.

Compared to Table IV, the results were improved a little bit. In particular with the best test accuracy achieved of 98.5% for the CNN. This corresponds to a 2.9% improvement over the 95.6% accuracy on the test set in the previous experiment. The efficiency of the LSTM also appears to have improved, as its accuracy on the test set increased by 3.7%.

The models are slightly slower due to their tuned model's additional computations, with now the CNN less quick on average per epoch than the LSTM, but this remains reasonable.

TABLE V: 50 Epochs Multinomial Tuned Models Recap

	window size	resolution (Hz)	train acc	train F_1	val acc	val F_1	train epoch time avg. (sec)
C	1 minute	10	98.69	98.69	97.43	97.53	170.98
N	**2 minutes**	**10**	**99.18**	**99.17**	**98.1**	**98.16**	**160.82**
N	3 minutes	10	98.81	98.83	97.94	97.96	155.26
L	1 minute	1	93.98	93.99	93.63	93.63	53.59
ST	2 minutes	1	95.35	95.35	93.83	93.87	46.86
M	**3 minutes**	**1**	**93.71**	**93.73**	**94.53**	**94.52**	**46.04**

In Figure 13, the leftmost graphs show that the overfitting identified with the baseline CNN model in Figure 9 no longer exists. Both the validation and training curves are much closer to each other.

In contrast, the line charts on the right reveal an instability in the learning of the modified LSTM model until the 15th epoch. From this point on, the curves stabilize a bit better until the end of the training.

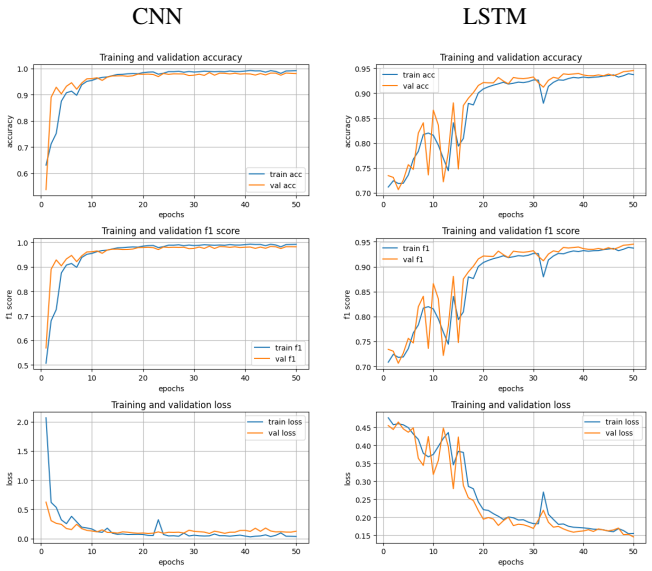

Fig. 13: Tuned multinomial models metrics plots

From the left confusion matrix in Figure 14, it can be observed that the ratio of valid instances classified as

awake and inversely is more consistent for the CNN than previously in the Figure 10. Concerning the right tuned LSTM confusion matrix, it looks more or less like the one in Figure 10, except that more valid and awake instances were correctly detected, increasing the model robustness.

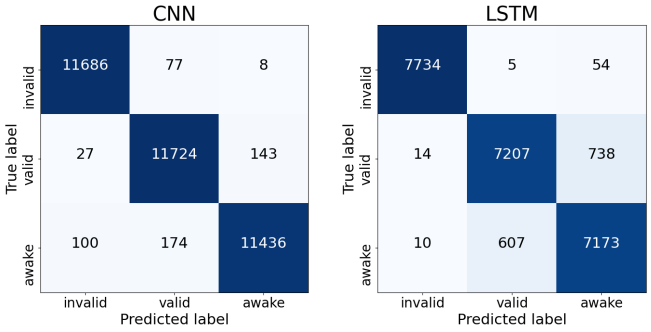

Fig. 14: Tuned multinomial models confusion matrices

The final confidence interval obtained for the best results of each model is as follows. 95% of the true classification will lie between 98.3% and 98.6% accuracy for the tuned CNN. Regarding the LSTM, 95% of the correct classification is likely to be between 93.6% and 94.2% accuracy.

V. Discussion

A. Results Interpretation

Firstly, it is shown in Table I that the computational time required to create the segmented subsequences from the entire time series dataset is more influenced by the total number of samples to be processed and then by the window size. The speed of the learning models is more influenced by the length of the samples to be processed. The lower the resolution of the signal, the faster the learning. Therefore it can be deduced that when learning a model, the choice of the size of the data set consists of finding a balance point between good and stable performances.

Secondly, considering Table II and Figure 7, it appears that using a balanced data set is a better option than leaving a minority class for training. The data augmentation technique used (SMOTE) has proven to be effective. Without it, the classifiers perform less well, especially for the multi-class problem. The main reason is related to the uneven distribution of labelling instances. Indeed, only 7.9% of the observations are identified as being of the awake type, which implies risks of bad prediction performances by the model, particularly considering that it is the most complicated class to identify.

Thirdly, the first baseline architectures designed for solving the problem have proven to be very efficient and complex enough to solve the binary classification task. While for the multinomial classification, the hyperparameters of the models could be further improved to make the architecture more suitable for the more complex problem and more robust.

Finally, looking at all the experiments on deep learning models, it can be observed that the classifiers have much less trouble classifying the invalid labeled class. One interpretation is that there is a considerable difference between the appearance of an invalid signal and that of a valid or awakening signal. Invalids are represented either by a flat signal, which means that one of the sensors is no longer measuring or if the mouth opening values are around or above 5 centimeters, which is physically declared impossible. Therefore, it is logical that this class is the easiest to recognize by the models and that the prediction results are very high. This also explains why the models learn very quickly from the train data in the binary classification and that after only 1 epoch, the models can already be considered trained. Concerning the likeness between valid and awake areas, the results indicate that in most cases, it is difficult to achieve as good a prediction accuracy as between the valid and invalid signal parts. This is certainly due to the similarity of these two types of signals, which can sometimes even be hard to differentiate by humans. Looking at figure 15 in the appendix, it can be seen that the only pattern that can help to differentiate them is in their behavior to be chaotic or not. If it can be seen that there are repeating patterns, it means that the patient is in a sleep zone or suffering from sleep apnea and not in wakefulness. Nevertheless, outstanding results have been achieved with the last experiment done with the tuned CNN.

B. Literature Results Comparison

The classification of human sleep signals is a popular area of research and many papers have been published on this topic. However, the majority of these studies are based on signal data containing more information, called Polysomnography[5] (PSG), and mainly deal with the classification of the different human sleep zones and not with the invalidity aspect of the data. The following paper [27] performed many experiments using a 19-layer CNN to try to differentiate the different stages of sleep by adding a new class one by one. The table VI is a summary of the results obtained. The first row is the only one of interest because the other classes are not covered in this work. In order to obtain a comparable test accuracy, a new result has to be computed from the left CNN confusion matrix in figure 14, considering only the valid and awake classes. For this new two-class classification setup, the best test accuracy obtained is 98.64%, compared to 98.33% in the first literature reviewed. This proves the reliability of the JAWAC signal for this type of classification compared to the PSG channel signals.

TABLE VI: Literature 1 Best Results Recap [27]

		model accuracy rate (%)		
classes	sleep stages	training	validation	testing
2	**Wake - Sleep**	**98.93**	**98.63**	**98.33**
3	Wake - {S1, S2, S3, S4} - REM	96.03	94.60	94.20
4	Wake - {S1, S2} - {S3, S4} - REM	92.92	90.86	91.39
5	Wake - S1 - S2 - {S3, S4} - REM	92.07	90.25	90.83
6	Wake - S1 - S2 - S3 - S4 - REM	90.01	88.32	89.51

[5]PSG monitors many bodily functions, including brain activity (EEG), eye movements (EOG), muscle activity or skeletal muscle activation (EMG), and heart rate (ECG), during sleep[26]

Then, another literature review is conducted [28], closer to the technique used for this research paper but still a bit far from the objectives of this study. Indeed, the aim of this second literature is to detect micro-awakenings on the same JAWAC signal technology. These are not exactly the same type as the one classified in this paper. They are more difficult to detect because they occur over a shorter period of time. These micro-arousals fracture sleep and impact its recovery process. Regarding the prediction results, the best performance obtained was an F_1-score of 75.49% using a CNN as shown in the Table VII. Relative to the results reported in this research paper, there are less precise but not totally comparable scores.

TABLE VII: Literature 2 Best Results Recap [28]

precision	recall	$\mathbf{F_1}$-score
84.35	68.32	75.49

C. Future Work

Although the results obtained are promising and the implemented models address the problem statement, advanced research on the classification of different target areas during sleep can still be considered. The following are some of the aspects that could be developed to push the performance further.

First, many other specific models variants architectures could be tested. Interesting new perspectives can still be explored, for example, with a new type of deep CNN called Inception Time or Echo State Networks, another sort of RNN that proposes new types of architectures [29].

Then, the complexity of the problem can be pushed even further. New classes capable of detecting the type of apnea event or the different human sleep stages must be added. This requires a more significant number of precisely labeled analyses to be added to the dataset.

Finally, future work should also address the optimization of the model's hyperparameters. Some potential improvements could be investigated, such as tuning the batch size, the model learning rate parameter and other static parameters during the experiments. Additionally, the implementation of an early stopping strategy designed to stop the training process when the validation accuracy stops increasing or the validation loss stops decreasing can be considered.

VI. Conclusion

This research is a first step in developing an efficient method for the automatic classification of time series validity. This work is unique because it attempts to meet the specific demands of an internship context. New neural network models are proposed and will be tested on the company's medical platform. The first task was to build a dataset on compressed time series files. The option of segmenting the time sequences into windows of the same size was chosen. In addition, 300 precisely labeled patient analyses were used due to biases in the core dataset and time constraints. The best results were obtained by using balanced data sets. Next, the second task was to develop a basic deep learning approach to solve the binary detection problem of valid and invalid data. Two options were proposed, one using a convolutional neural network model and the other using the long-term memory algorithm, a specific recurrent neural network architecture. The basic models of these two learning algorithms achieved, after 5 epochs, test accuracy and F_1-score of approximately 99.5% for the CNN and test accuracy and F_1-score of 99.8% for the LSTM. Then, the third task was to increase the complexity of the problem by adding a third cluster to classify the awake regions in the patient's sleep. The same models are kept to handle this new problem. After 50 epochs, the the CNN achieved an accuracy and F_1-score of approximately 95.6% on the test set and 90.2% for the LSTM. This new task proved to be more complex than the two-class classification. Model tuning improved the results over their original version. The final accuracy and F_1 prediction results were 98.5% for the CNN and 93.9% for the LSTM. Avenues for future research to improve the model were suggested in the discussion, including the use of more accurately labeled data.

References

[1] Nomics Sleep Care. *About us*. URL: https://www.brizzy.eu/about-us/.

[2] J. Brownlee. *1D Convolutional Neural Network Models for Human Activity Recognition*. 2018. URL: https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/.

[3] J. Brownlee. *LSTMs for Human Activity Recognition Time Series Classification*. 2018. URL: https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/.

[4] D. Tuan Pham. *Time–frequency time–space LSTM for robust classification of physiological signals*. 2021. URL: https://www.nature.com/articles/s41598-021-86432-7.pdf.

[5] A. Deckard. *Sleep Apnea Statistics*. URL: https://cpapsupplies.com/blog/sleep-apnea-statistics.

[6] A. Amidon. *A Brief Survey of Time Series Classification Algorithms*. 2020. URL: https://towardsdatascience.com/a-brief-introduction-to-time-series-classification-algorithms-7b4284d31b97.

[7] M. Granat. *How to Use Convolutional Neural Networks for Time Series Classification*. 2019. URL: https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57.

[8] F. Edin. *Time series classification – an overview*. 2020. URL: https://developersbay.se/time-series-classification-an-overview/#F1.

[9] D. Alvarez-Estevez. *European Data Format*. URL: https://www.edfplus.info.

[10] *MNE-Python Development*. URL: https://mne.tools/stable/overview/development.html.

[11] iTech. *The Importance of Data Processing in Machine Learning*. 2020. URL: https://itechindia.co/blog/the-importance-of-data-processing-in-machine-learning/.

[12] *About pandas*. URL: https://pandas.pydata.org/about/index.html.

[13] S. Dieter. *Stuck in the middle – mean vs. median*. 2017. URL: https://www.clinfo.eu/mean-median/.

[14] J. Brownlee. *SMOTE for Imbalanced Classification with Python*. 2020. URL: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.

[15] S. Satpathy. *Overcoming Class Imbalance using SMOTE Techniques*. 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

[16] Y. Kim. *Convolution Neural Networks for Sentence Classification*. 2014. URL: https://arxiv.org/abs/1408.5882.

[17] P. Sharma. *Keras LSTM Layer Explained for Beginners with Example*. 2021. URL: https://machinelearningknowledge.ai/keras-lstm-layer-explained-for-beginners-with-example/.

[18] AC. Anderson. *Understanding LSTM Networks*. 2017. URL: https://stackoverflow.com/q/44273249.

[19] K. Chowdhury. *10 Hyperparameters to keep an eye on for your LSTM model — and other tips*. 2021. URL: https://medium.com/geekculture/10-hyperparameters-to-keep-an-eye-on-for-your-lstm-model-and-other-tips-f0ff5b63fcd4.

[20] S. Ramesh. *A guide to an efficient way to build neural network architectures- Part I: Hyper-parameter selection and tuning for Dense Networks using Hyperas on Fashion-MNIST*. 2018. URL: https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-i-hyper-parameter-8129009f131b.

[21] Y. Tang et al. *TensorFlow*. URL: https://github.com/tensorflow/tensorflow.

[22] F. Chollet et al. *Keras: Deep Learning for humans*. URL: https://github.com/keras-team/keras.

[23] P. Huilgol. *Accuracy vs. F1-Score*. 2019. URL: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2.

[24] J. Brownlee. *Confidence Intervals for Machine Learning*. 2018. URL: https://machinelearningmastery.com/confidence-intervals-for-machine-learning/.

[25] C. Maklin. *Dropout Neural Network Layer In Keras Explained*. 2019. URL: https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab.

[26] Wikipedia. *Polysomnography*. 2021. URL: https://en.wikipedia.org/wiki/Polysomnography.

[27] O. Yildirim, U. Baran Baloglu, and R. Acharya. *A Deep Learning Model for Automated Sleep Stages Classification Using PSG Signals*. 2019. URL: https://www.mdpi.com/1660-4601/16/4/599.

[28] P. Hockers. *Application of modern machine learning techniques to the detection of micro-awakenings on a midsagittal jaw motion signal*. 2021. URL: https://matheo.uliege.be/bitstream/2268.2/11239/4/TFE_Report_Final_Pierre_Hockers.pdf.

[29] M. Del Pra. *Time Series Classification with Deep Learning*. 2020. URL: https://towardsdatascience.com/time-series-classification-with-deep-learning-d238f0147d6f.

APPENDIX

A. Classifier Invalid Output Graph

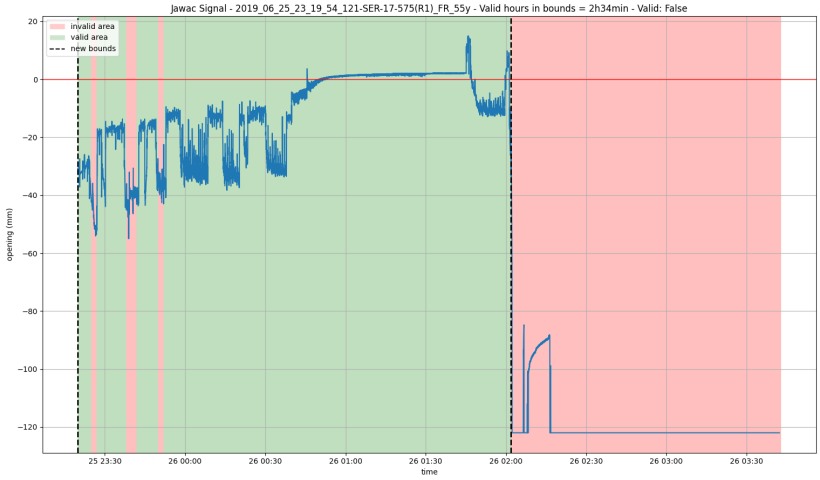

Fig. 15: Invalid output analysis example

Figure 15 is an example of time series classified with the binary LSTM model, where regions of invalidity are represented by red labeling. In fact, the areas where the signal is flat (from 2:15 onwards) or where the recorded values are around or above the possible limit (5 centimeters of mouth opening), will be classified as invalid. In this case, it can be observed in the graph title that there are only 2 hours and 34 minutes of valid signal in the new bounds proposed by the algorithm. This is far below the recommended 4 hours, so the patient will have to retest to receive a diagnosis.

B. Classifier Valid Output Graph

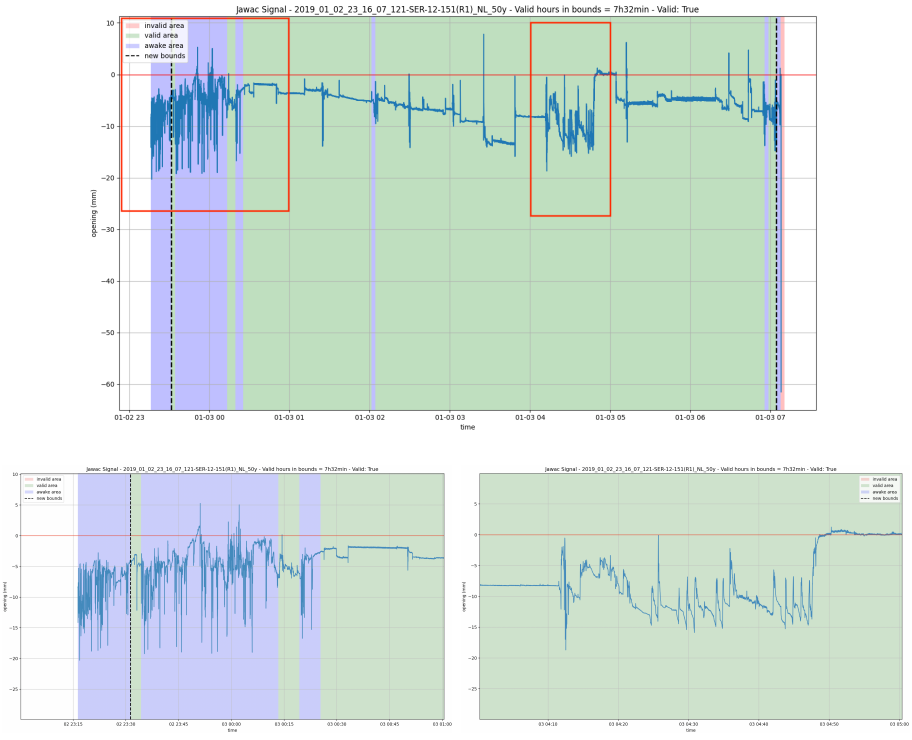

Fig. 16: Valid output analysis example

The second Figure 16 represents an example of valid analysis classified with the multinomial CNN model. In fact, the two red rectangles are zoomed in on the lower left and right graphs, respectively, showing a clear distinction between an awake and an apnea event region. The main feature to distinguish them is the chaotic way the mouth opening is represented when the patient is no longer sleeping. The figure on the bottom right contains repetitive patterns, which means that the patient certainly does have sleep apnea. In this case, the analysis is preserved for diagnosis since 7 hours and 32 minutes are detected as valid.