Mini-Project (ML for Time Series) - MVA 2023/2024

Matteo Marengo matteo.marengo@ens-paris-saclay.fr
Hugo Robert hugo.robert@ens-paris-saclay.fr

December 18, 2023

TSFEL: Time Series Feature Extraction Library

1 Introduction and contributions

1.1 Scientific context of the article

The article that we are studying [2] introduces a **Time Series Feature Extraction Library**. Indeed, extracting features from a time series dataset is one of the most challenging tasks when designing and solving a machine learning pipeline for classification. One can extract features from a dataset by being a specialist in the field from which the time series dataset has been designed (e.g for EEG measurements to assess epilepsy, it is interesting to extract Signal amplitude average, Signal amplitude standard deviation, symmetry, power spectral density and signal curve length [4]). However, these feature extraction steps are often tedious and require proficiency. Therefore, TSFEL or other feature extraction libraries try to overcome this challenge by extracting a fast exploratory analysis and an automated process on a multidimensional time series. Therefore the question that we will try to answer through this report is: **Are these solutions a leap forward to design ML pipelines or do they only mix the things even more?**

1.2 Task to be solved

Therefore, after defining this problem, we decided to focus our work on two aspects of the TSFEL library.

1. To what extent does extracting generic features followed by a selection step allow addressing a wide range of problems? This will be contrasted with carefully chosen and custom-built features.
2. What are the limitations of the TSFEL library? What are the areas for improvement?

To do that, a benchmark will be made on four datasets **(Walk DTW, Radars, ECG, EEG)** comparing four features extraction method **(custom-built features, tsfel, tsfresh, cesium)** to see in which cases the classification accuracy is the best.

1.3 General Information

The four datasets we are using are the Human Locomotion with Inertial Measurements Units [8], the Radars Dataset (threat or not threat), the EEG for epilepsy detection [1], the ECG for abnormal signal detection [6]. The three feature extraction libraries we are comparing are: TSFEL [2], CESIUM and TSFRESH [3]. To use these available libraries, we will need to adapt the feature extraction pipeline and use the most relevant metrics for feature selection.

Finally, concerning the repartition of work between the two students, Matteo has mainly looked at ECG and EEG datasets, doing the comparisons within the libraries, whereas Hugo was more focused on DTW for walk and Radar datasets, doing many tests to select the best data each time. Both students have worked equally on the report, from the literature review to the writing part.

2 Method

2.1 TSFEL Features Extraction pipeline

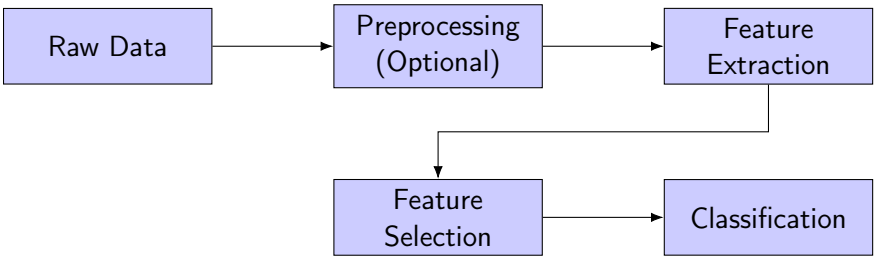
Here we are going to describe the features extraction pipeline of TSFEL. First, the implemented feature methods are realised with Numpy and SciPy [2]. The TSFEL features can be

grouped into three categories: **temporal, statistical and spectral**. Something that is done with that library is the fact that there is preprocessing and windowing. Indeed, the time series that are being studied are divided into time windows to extract some of the features. The window size and sampling frequency can be specified according to the dataset we used.

Once the features have been extracted, they are then filtered to keep only the most relevant ones. The generic metrics commonly used to define whether a feature is relevant are the correlation between features (to remove features that are correlated) and the variance (to remove less relevant features). Of course, other metrics can be used to reduce the dimensionality of the data, such as PCA or methods based on statistical tests (chi-square, Pearson independence, etc.). TSFRESH and CESIUM Features Extraction pipeline are briefly explained in the Appendix.

2.2 Time Series Classification Pipeline

Let's present the typical time series classification pipeline that will be used to do classification.



This study aims to compare four different methods of extracting features using a benchmark on different datasets. The first is based on the traditional technique of manually understanding and interpreting the data to extract a limited number of features containing the information in the time series. Most of the time series analysis work is carried out in the "pre-processing" section, where expertise is required to define the most relevant characteristics. This technique aims to define a reference level.

The other three methods use three different Python libraries (TSFEL [2], CESIUM and TSFRESH [3]) to extract a large number of generic features from the time series before being selected to train our models only on the most relevant ones. Little work is therefore done in pre-processing while the main focus is on feature extraction and selection.

To compare the four methods, we will use the extracted features to create a classification method using the model best suited to our situation, and we will record the accuracy obtained. This will be indeed a broad subject of discussion as to use a KNN, a DecisionTree or a Neural Network has an important impact on the final result depending on the input data.

3 Data

Here we are describing the dataset we are using and based on expert-domain knowledge what should be the required features to extract?

3.1 Human Locomotion

This dataset is adapted from the first practical. It consists of signals collected with inertial measurement units from 230 subjects that follow a fixed protocol [8]. The task is to classify them between healthy and non-healthy patients. There we will compare classification with the DTW Algorithm and the feature extraction techniques.

3.2 Radars

In this dataset, we intercept radar signals and then try to determine the type of source (a "threat" or "non-threat" radar in a military context). To do this, the dataset comprises a series of samples. Each sample represents a series of pulses received. These pulses were then pre-processed to extract 5 pieces of information, each of which is recorded in a dedicated time series: the duration of the pulse, its start date, its power, its frequency and its theta and phi angles (corresponding to the direction in which the pulse was received).

Using this dataset, we determined whether 'generic' features could be adapted to this type of data. To compare the results, we also trained a model on more specific features such as the number of local minimums in power, weight and height of the largest lob in power, as well as an estimate of the pulse sending frequency. For each method, the goal was to train a random forest based on the given features. From this, we will measure the obtained accuracy.

3.3 EEG for epilepsy detection

The Dataset that is used here is a EEG dataset [1] where data are splitted between three classes; **normal (Z,O), interical (N,F), ictal(S)**. A physiological analysis of this data comes from [4]. The interictal period is when an epilepsy patient does not have an overt seizure, but EEG signals may show abnormalities like spikes or sharp waves, indicating a seizure predisposition. In contrast, the ictal period occurs during an active seizure, where the EEG typically reveals highly abnormal brain activity patterns. Five standards features for EEG analysis [4] are therefore **mean_signal, std_signal, mean_square_signal, abs_diffs_signal and skew_signal**. An other idea is to use the previous defined features but each signal is decomposed into five frequency bands using a discrete wavelet transform as suggested in [7].

3.4 ECG for classification of Normal, Arrhythmia and Congestive Heart Failure

The Dataset that is used here is an ECG Dataset [6] and the code is extracted from [5]. The authors explained that frequency and time-frequency domain features by applying fast Fourier transform and wavelet transforms should be extracted features. The classification is done with a feed-forward neural network. The dataset consists of 162 ECG recordings and diagnostic labels sampled at 128 hertz.

4 Results

4.1 Benchmark presentation and Analysis

4.1.1 Human Locomotion

To compare DTW algorithms and Features Extraction libraries we used the kNN method to predict the label of a time series based on its similarity to the training data. Results obtained are

depicted in Fig 1.

Method	Accuracy
DTW	0.826923
TSFEL	0.711538
CESIUM	0.634615
TSFRESH MINI	0.596154
TSFRESH EFFICIENT	0.653846
TSFRESH COMPREHENSIVE	0.653846

Figure 1: Results Comparison of various features extraction techniques vs DTW on the Human Locomotion Dataset.

The results obtained in this dataset show that extracting generic features does not produce results as good as those obtained by extracting a single 'expert' metric such as DTW. However, the TSFEL library seems to be by far the library that gives the best results. It should also be noted that all libraries take approximately the same time to extract features (between 20 and 30 seconds).

4.1.2 Radars

For this dataset, we used the random forest model. Results obtained are depicted in Fig 2.

Method	Accuracy
personalized	0.88375
TSFEL	0.90250
CESIUM	0.89625
TSFRESH MINI	0.67000
TSFRESH EFFICIENT	0.77125

Figure 2: Results Comparison of various feature extraction techniques on the Radar Dataset.

As we can see, in this dataset the extraction of a large number of generic features followed by a generic selection of these features (only based on correlation and variance) gives better results than a model train with personalized features. This result can be explained by the fact that the specialized features are not very complex, which probably means that the information contained in these features can also be extracted through a combination of other more generic features. The bad results for TSFRESH might however be explained by two things; either our implementation has an error, either having selected too many features introduces too many variances and the lack of explainability.

4.1.3 EEG for epilepsy detection

Let's compare classification results for features extracted with [4] method, [7] method, TSFEL [2], Cesium, TSFRESH [3]. We extract features and then to classify we use 3 NN classifiers for [4] and [7] methods as described in [4] and Random Forest Classification for the others. The results are shown in Fig 3. A bar chart of these results is also shown in Fig 5.

	Training Accuracy	Test Accuracy
Cesium	1.000000	0.832
TSFEL	1.000000	0.968
TSFresh	1.000000	0.944
TSFresh Mini	1.000000	0.760
Guo et al.	0.928000	0.832
Wavelet Transform	0.978667	0.952

Figure 3: Results Comparison of various feature extraction techniques on the EEG Dataset.

The dataset shows that the Guo et. al method and Cesium yield identical results due to their extraction of a limited feature set. TSfresh Mini, extracting only a few generic features not specifically relevant to the study, performs the worst. Conversely, TSFEL, TSFRESH Efficient, and Wavelet Transform achieve similarly high accuracies (0.968, 0.944, 0.952), indicating that selecting the right features to extract, as suggested by [7], can lead to high accuracy with lower computational time compared to the other methods.

4.1.4 ECG for classification of Normal, Arrhythmia, and Congestive Heart Failure

Frequency features from ECG recordings are extracted using Fast Fourier Transform and time-frequency features through wavelet transform. These features are processed by a feed-forward neural network, achieving a high validation accuracy of 0.9573. However, using TSFEL for feature extraction before running the neural network reduces the accuracy significantly to 0.3057, which is almost random for a three-class problem. The same low accuracy is observed with the Cesium method, indicating unexpected performance outcomes.

Testing various hypotheses on ECG datasets revealed that using the full dataset with TSFEL yields an accuracy of 0.6038. Switching to a RandomForest classifier instead of a neural network significantly improves accuracy to 0.856, highlighting the importance of classifier selection. Similarly, using Cesium and RandomTree results in a 0.976 accuracy. With TSFRESH, neural networks only achieve a 0.307 accuracy, while RandomForest reaches 0.976. These results suggest that neural networks are better suited for large data sets, like scaleograms with 195075 input parameters, but for smaller feature sets, machine learning classifiers like RandomForest are more effective.

4.2 Conclusion

As we have seen from the datasets studied, the relevance of using generic features needs to be considered on a case-by-case basis. Expertise is therefore always required to study a given dataset. However, the use of libraries such as TSFEL is beneficial when it comes to extracting certain features quickly. In some cases, their use enables us to obtain results similar to those obtained using 'expert' features in a much shorter time, while in other cases their performance is limited by the nature of the time series being studied.

Systematic use of this type of library can produce initial results in a minimum of time. On the other hand, limiting oneself to the automated extraction of generic features can be detrimental to the analysis, as it does not provide an in-depth understanding of the dataset being studied. It would also be interesting to include in our benchmarks a comparison with new libraries that we discovered during our tests, such as the Sktime library.

References

[1] Rieke C Mormann F David P Elger CE Andrzejak RG, Lehnertz K. Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. In *Phys. Rev. E, 64, 061907*, 2001.

[2] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[3] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

[4] Ling Guo, Daniel Rivero, Julián Dorado, Cristian R. Munteanu, and Alejandro Pazos. Automatic feature extraction using genetic programming: An application to epileptic eeg classification. *Expert Systems with Applications*, 38(8):10425–10436, 2011.

[5] Tanveer Khan. Time series based feature extraction: Electrocardiogram (ecg) data. 2021.

[6] Antônio H. Ribeiro, Manoel Horta Ribeiro, Gabriela M. M. Paixão, Derick M. Oliveira, Paulo R. Gomes, Jéssica A. Canazart, Milton P. S. Ferreira, Carl R. Andersson, Peter W. Macfarlane, Wagner Meira Jr., Thomas B. Schön, and Antonio Luiz P. Ribeiro. Automatic diagnosis of the 12-lead ECG using a deep neural network. *Nature Communications*, 11(1):1760, 2020.

[7] Abdulhamit Subasi. Automatic recognition of alertness level from eeg by using neural network and wavelet coefficients. *Expert Systems with Applications*, 28(4):701–711, 2005.

[8] Charles Truong, Rémi Barrois-Müller, Thomas Moreau, Clément Provost, Aliénor Vienne-Jumeau, Albane Moreau, Pierre-Paul Vidal, Nicolas Vayatis, Stéphane Buffat, Alain Yelnik, Damien Ricard, and Laurent Oudre. A Data Set for the Study of Human Locomotion with Inertial Measurements Units. *Image Processing On Line*, 9:381–390, 2019. https://doi.org/10.5201/ipol.2019.265.

5 Appendix

5.1 TSFEL: feature extraction pipeline

As described in [2], TSFEL ambition was to help to extract features from multidimensional time series by an automated process. TSFEL works like that: time series are passed as inputs for the main TSFEL extraction method. There is the windowing step where the time series are divided between windows from where features will be extracted. If there is one figure to bear in mind for TSFEL it is this one, cf Fig 4.

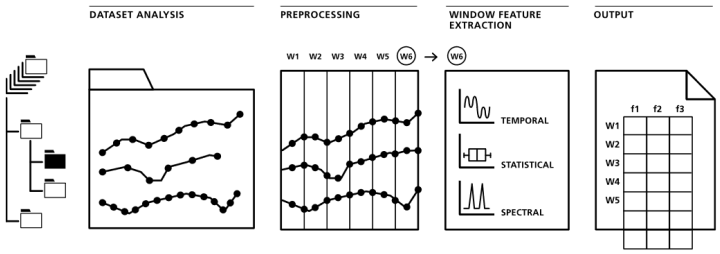


Figure 4: TSFEL process to extract features Adapted from [2].

An important part of TSFEL and other feature extraction libraries are the feature selection and some tasks have therefore to be done:

```
# Highly correlated features are removed
corr_features = tsfel.correlated_features(X_train)
X_train.drop(corr_features, axis=1, inplace=True)
X_test.drop(corr_features, axis=1, inplace=True)

# Remove low variance features
selector = VarianceThreshold()
X_train = selector.fit_transform(X_train)
X_test = selector.transform(X_test)

# Normalising Features
scaler = preprocessing.StandardScaler()
nX_train = scaler.fit_transform(X_train)
nX_test = scaler.transform(X_test)
```

Listing 1: Python example to select features

TSFEL additionally provides the computational time for each feature, it is important to know for a data scientist as it would provide him insights on which features are relevant depending on the task he has to do. An improvement method that should be led is the introduction of a nonlinear method that does not exist in TSFEL.

5.2 CESIUM: feature extraction pipeline

Cesium is a basic feature extraction pipeline as it only extracts features that the user requests, there are no lazy options such as the one in TSFEL. However, it leads to a reduced computational time.

5.3 TSFRESH: feature extraction pipeline

TSFRESH is a high-level feature extraction library. It includes nonetheless scikit-learn compatible transformers such as FeatureAugmenter and FeatureSelector (to extract the features and then select them). When doing the feature extraction, we can either select all features without parameters and all features without parameters, it takes a lot of time to run. We can also extract only the efficient ones where high computational features are removed. Finally the minimal setting is set to handle quick tests.

5.4 EEG Dataset Results

Here is (Fig 5) the representation of the EEG Dataset classification accuracy with the different features extraction methods.

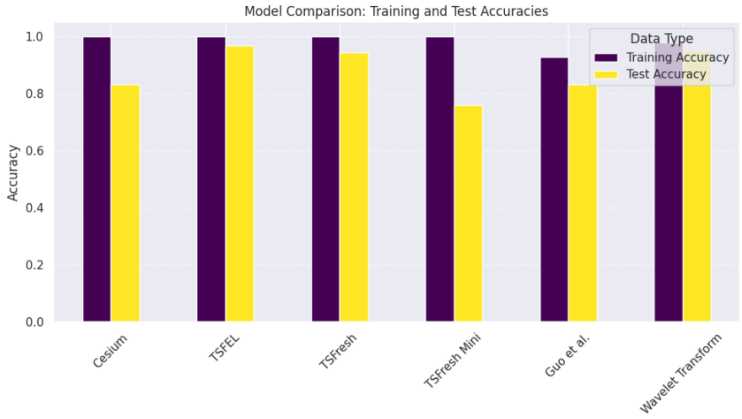


Figure 5: EEG Results Bar Chart Comparison

5.5 ECG Dataset Results

5.5.1 Features extracted

In Fig 6, we can see the three different ECG signals that we want to classify between Normal, Arrhytmia, and Congestive Heart Failure.

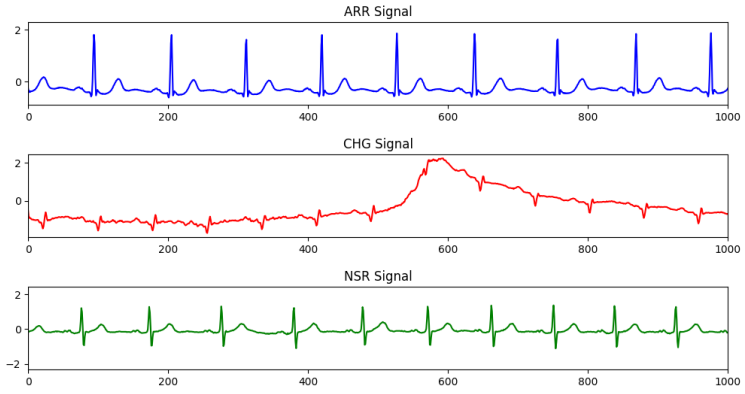


Figure 6: Comparison of the three classes of signals in ECG.

Here is the representation of the scaleograms extracted from the three classes (cf Fig 7). As a recall, scaleogram is a visual representation that shows how the scale of a wavelet transform varies with time. It is very useful to analyze non-stationary signals, where the frequency content varies with time.

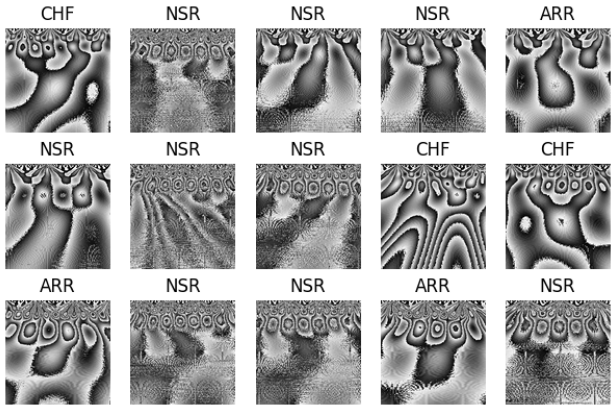


Figure 7: Comparison of the scaleograms for the three different classes

5.5.2 Classification results with FFT and Wavelet Features

With these features extracted, the accuracy obtained is high and the loss strongly diminishes proving that the features extracted are the correct ones (cf Fig 8).

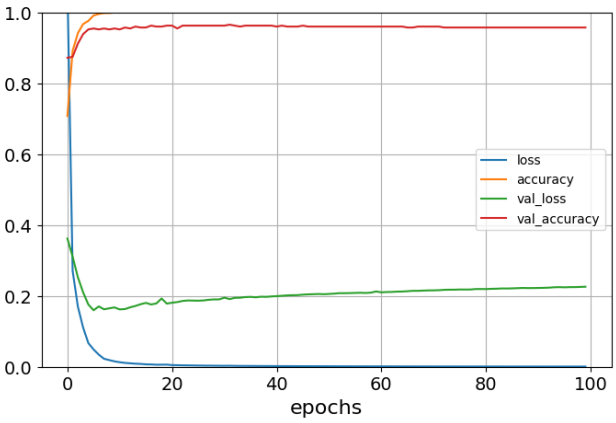


Figure 8: Training and Validation loss/accuracy with FFT and Wavelet Features.