POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis in collaboration with Tierra S.p.A.

Profile heavy duty vehicle usage based on CAN bus data mining

Supervisors:

Prof. Francesco Vaccarino

Prof. Luca Cagliero

Company tutors:

Lucia Salvatori

Riccardo Loti

Candidate:

Silvia Buccafusco

October 2020

Summary

In this thesis work a real case problem concerning heavy duty vehicles' usage patterns identification is addressed.
Even if in the literature there are several cars usage patterns identification, the same kind of analysis is less frequently carried out on industrial, construction or off-roads vehicles. However, thanks to the wide spread of IoT devices and the firmly established cars connected mobility, the heavy-duty in-vehicle connectivity is growing in importance.
To this purpose, multivariate analysis of multiple CAN signals techniques based on clustering and patterns discovery from time series data is presented. At first, ultra-fine, asynchronous and heterogeneous signals have been analysed: the relevant parameters to be monitored have been identified, the most appropriate level of aggregation of data has been suggested and series characterized by different sampling rates have been properly combined. Then, a multivariate time series segmentation strategy based on an application of the VALMOD algorithm has been proposed. Finally, different clustering and patterns discovery methods are presented, inspecting signals properties both in time and frequency domain.
The results of the proposed procedures have been finally evaluated applying them to a real use case: three different usage patterns have been identified, respectively corresponding to idle, moving or regular working and higher workload. The results have been validated both by domain experts and by means of the additional information provided by NMEA 0183 messages data.
In conclusion, an autoencoder-based deep learning for multivariate time series clustering is presented to inspect the presence of hidden features that may have not been considered before.

Table of Contents

List of Tables

List of Figures

Acronyms

ACF

Autocorrelation Function

ACVF

Autocovariance Function

AIC

Akaike Information Criterion

ANN

Artificial Neural Network

BIBO

Bounded-Input Bounded-Output

BIC

Bayesian Information Criterion

CAN

Controller Area Network

CAN FD

Controller Area Network Flexible Data-Rate

CCE

Categorical Cross Entropy

DC

Direct Current

DFT

Discrete Fourier Transform

DTFT

Discrete-Time Fourier Transform

FPE

Akaike's Final Prediction Error

HQIC

Hannan-Quinn Information Criterion

IDTFT

Inverse Discrete-Time Fourier Transform

IoT

Internet of Things

ISO

International Organization for Standardization

MSE

Mean Square Error

MVAR

Multivariate AutoRegressive model

NMEA

National Marine Electronics Association

OBD2

On-Board Diagnostics II

PACF

Partial Autocorrelation Function

PGN

Parameter Group Number

SAE

Society of Automotive Engineers

SPN

Suspect Parameter Number

VALMOD

Variable Length Motifs Discovery

Acknowledgements

Vorrei ringraziare il Professor Francesco Vaccarino e il Professor Luca Cagliero per l'attenzione che hanno dimostrato nei miei confronti durante lo svolgimento di questa tesi, per esser stati una costante fonte di stimoli, di motivazione e di ottimismo, per aver sempre trovato un momento per confrontarci e per avermi guidato fino alla conclusione di questo elaborato.

Un sentito ringraziamento va all'azienda Tierra S.p.A., in particolare a Lucia Salvatori, Riccardo Loti e Calogero Carrabbotta, per avermi consentito di portare a termine questo percorso di formazione in modalità smart working, superando le difficoltà della distanza offrendomi costante disponibilità, supporto, professionalità ed esperienza.

Ringrazio i miei genitori per aver reso possibile questi cinque anni a Torino, accettando che abbia deciso di iscrivermi "all'unico corso di laurea non presente nei cataloghi dalle università della tua città". Un grazie alla mia famiglia per esser stata un sostegno costante e presente, per aver creduto in me, per aver incondizionatamente supportato ogni mia decisione. Un grazie in particolare al mio papà per aver risposto al telefono a qualsiasi ora, prima di ogni esame al mio grido "mi bocciano!" e per non averci creduto mai, per avermi insegnato ad accettare serenamente anche i miei limiti, nella consapevolezza e tranquillità di aver fatto tutto quel che era in mio potere.

Grazie a Mattia per aver camminato al mio fianco passo dopo passo, per aver condiviso ogni momento di questo percorso, per avermi capita, appoggiata, motivata, per essere stato una ventata di spensieratezza in grado di portare una risata e un raggio di sole nei miei momenti più neri.

Grazie ad Amedeo, per essere stato il punto fermo nel mio disordine.

Ad Antonio, perché "senza di te Torino sarebbe stata più grigia".

Alle persone che sono rimaste al mio fianco in questi anni, che conoscono quella che ero e quella che sono e che mi abbracciano un po' più forte ad ogni mia ripartenza: a *os amigos de tubarão que ta*, per avermi fatto sentire a casa ad ogni mio ritorno a Roma, per aver quasi fermato il tempo all'ultimo giorno di scuola. A Elena per l'affetto, la dolcezza e il suo incurabile ottimismo. A Costanza per aver capito e assorbito la mia rabbia, per l'ironia e la complicità.

Agli UCCM per la creatività, la spontaneità e la fantasia: a Checco, per essere stato il mio "coinquivicino" e avermi curato l'anima cucinando polpette e carbonare, a Fraf per esser stato fonte certa e inesauribile di libri, dispense e fastidiose domande con poche e faticose risposte, a Chiare per avermi dato un motivo per essere felice di tornare a casa ogni sera e per farmi sentire, ogni tanto, la mancanza di quel piccolo rifugio di follia. A Giulione e Matteo e il "club del 99", per la nostra immancabile autoironia. A Rosy, per avermi sopportato in una settimana di musei a Paris e aver applaudito, in ognuno di questi, le mie puntualissime opere d'arte. A Leo, per essere stato il mio compagno di pause caffè. A Stefano per aver sopportato cinque anni di battute e risposte acide, senza prendermi sul serio (quasi) mai. A Simone, per aver riempito i miei vuoti e non avermi mai fatta sentire sola.

Grazie per aver fatto parte di questo mio percorso di crescita.

SB

Chapter 1

Introduction and relative work

Machine Learning is defined by [1] as *"the automated detection of meaningful patterns in data"*.
It can be seen as a branch of Artificial Intelligence, even if the aim of Machine Learning is not process automation, but use patterns found in data to understand them, predict future data or any response variable of interest or provide support to decision processes under uncertainty [2]. In this sense, Machine Learning is a set of tools which act as a complement to human intelligence.

Nowadays, because of digitization and its consequent fast large size production of data, Machine Learning has become one of the most important areas of Computer Science and it is often associated with a huge amount of information, commonly called *Big Data*.
However, Machine Leaning is an interdisciplinary field with features shared with mathematics, data analysis, statistics and information theory [3].

A Machine Learning algorithm is based on automatic learning: it takes as input the data that can be seen as *experience*, and transform them into an output that can be of different forms, such as a response variable or another algorithm. The meaning of *learning* is that this output can be interpreted as *knowledge* extracted from experience [1].

Commonly, two different scenarios can be considered.
The former is *supervised learning*: it is also called predictive learning, because the aim is to find a model basing on the available data to explain the underlying relationship between input and output variables in order to make predictions on the unknown response variable of new observations.
Under this scenario, a *training set* is given, a finite set composed by n items or

observations, also called *predictors*, each associated with a response variable. Examples of supervised algorithms are linear regression and logistic regression. A different scenario is *unsupervised learning*: it is the case when for each observation of input data there is a set of measurements but no response variable. If this is the case, a descriptive analysis can be performed and this process is also known as *knowledge discovery* because its main purpose is understanding data [2]. Examples of unsupervised algorithms are clustering or principal component analysis.

In this thesis work, an unsupervised clustering problem is addressed in order to discover heavy duty vehicles' usage patterns and workload status from CAN bus data. Even if in the literature there are several cars usage patterns identification, the same kind of analysis is less frequently carried out on industrial, construction or off-roads vehicles. However, thanks to the wide spread of IoT devices and the firmly established cars connected mobility, the heavy-duty in-vehicle connectivity is growing in importance. It is extremely of interest for companies being able of monitoring their equipment in order to optimize maintenance, production, business and investments and a support to decisions in this sense can be obtained analysing data generated from the large amount of sensors installed nowadays on each type of vehicle.
Data are collected by a Z55 device: it is a data logger provided by Tierra S.p.A., a company operating in the IoT sector internationally recognized for providing to their customers sophisticated and reliable telematics solutions for management, maintenance and remote diagnostics of equipment. This thesis was developed as a result of my internship in Tierra S.p.A. and it is part of the applied research and data analytics collaboration of the company with the SmartData@PoliTO center for Big Data and Machine Learning technologies.

The company provides their clients with an on-board device with a SIM to collect CAN bus data, then transmitted and stored inside Tierra cloud infrastructure. Recorded data can be visualized and managed by the clients by means of a customized web-based remote management systems.

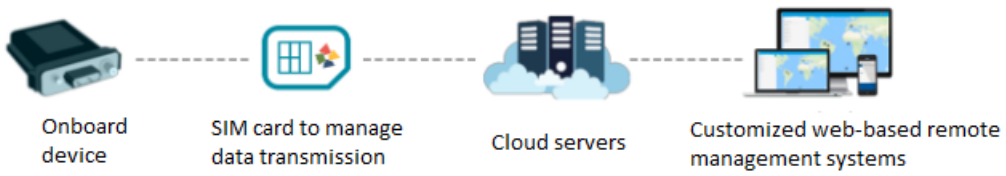

Figure 1.1: Tierra solution
Images from www.tierratelematics.com

CAN bus data are generated in the vehicle at high frequency and gathered by a controller. However, only an aggregate report obtained computing one statistic (max, min, average or last) every 10 minutes is sent to the centralized server [4]. This means that each line of the database corresponds to a value for a single SPN that summarizes its behaviour during a time window of about 10 minutes. For the moment, this type of information can be checked in real-time by the customers, together with some statistics computed on these values, without applying any advanced algorithm. For example, since Tierra identifies some different possible workloads for the monitored vehicles, one of the statistics provided to their clients is the total amount of working time spent in each workload type. However, whenever the vehicle is in a given state is determined through some manually set thresholds, driven by domain experts. The idea of this thesis is to apply some advanced machine learning techniques in order to identify different usage patterns, phases and working states from time series data, improving the provided analysis to the clients.
Furthermore, since there is no particular evidence to set the data granularity to 10 minutes, for the following analysis non aggregate data are used. In this way it is possible to evaluate improvements (if any) and set a more appropriate data granularity as result of a trade off between clients and company specifications, accuracy and computational costs.

The thesis work is organized as it follows.
In *chapter two* CAN bus data exploration is performed. After a brief introduction to CAN data and relative standards, a descriptive analysis of the dataset and of the main data cleaning steps is carried out. Some preliminary statistics are computed and a deep description of parameters under analysis is provided. Finally, the procedure identified for detecting working cycles is described.

In *chapter three* the problem of synchronization of signals is addressed. A technique based on signal processing analysis and Fourier transform is applied, since each parameter measurements can be seen as a digital signal. The purpose of this procedure is to obtain constant rate and synchronized measurements in order to interpret each parameter as a component of a multivariate time series.

Basing on the results of the previous chapter, the main tools for time series analysis are applied in *chapter four*. At first, the measurements associated with each parameter are considered in isolation, performing a univariate analysis of stationarity, trend, seasonality, autocorrelation and partial autocorrelation. Then, interactions between signals are analysed by means of cross-correlation, Pearson's correlation and Granger causality. Thanks to results of this section, feature selection is performed in order to identify the meaningful parameters for the following analysis.

In *chapter five* two different clustering techniques are applied to identify common usage patterns. The former, namely clustering by values, exploits the time series synchronization, considering each data point as an item to cluster but without taking into account the temporal order of time series data. As result, it provides some automatic thresholds to define the vehicles' states, of the same kind of the manually set ones used by the company. The second clustering strategy presented instead is based on the common technique used for unsupervised time series clustering consisting in creating features to use as input to the algorithm. However, in order to detect vehicles' states basing on the shape of the time series data, features are computed in the frequency domain, describing peaks, powers and fundamental frequencies in three distinct bands, separately for high, medium and low frequencies. The first step of this technique is time series segmentation, performed as an application of the VALMOD algorithm, used to discover repeated patterns in data.
Combining the results of both cluster techniques, it is possible to obtain a correct identification of the expected workloads.
In the last part of the chapter, the obtained workloads types have been deeply analysed thanks to the additional information provided by NMEA 0183 messages data, used to validate and further interpret results.

Finally, in *chapter six* an autoencoder-based deep learning technique for time series data clustering is presented. The aim of the described method is to validate the obtained results and to highlight the presence of hidden features in data by inspecting the properties of the items assigned to different classes by the two clustering method. From this kind of analysis is possible to identify some strategies to further improve results in future works.

Chapter 2

Controller Area Network Bus Data Analysis

2.1 Introduction to Controller Area Network bus data

Controller Area Network (CAN) is a message-based legacy protocol for in-vehicle data communication, commonly used in automotive industry and embedded systems networking. It was invented by Robert Bosch in 1986 to allow faster and robust serial communication between microcontrollers, overcoming the inefficiency of separately connecting each other by means of a broadcasting communication technique. The most relevant CAN specification, CAN 2.0, was published by Bosch in 1991 and thanks to its widespread popularity, it became in 1993 the international standard ISO 11898 [5]. Since then, several higher-level protocols have been standardized on CAN [6].

Nowadays, each type of vehicle is equipped with a large amount of sensors capable to capture high frequency generated CAN messages, that are gathered by a controller and then collected and processed. Even if CAN protocol was first created for automotive applications, several other industries adopted CAN for a wide variety of applications, ranging from medical devices to aerospace contexts. Some of the most common standards are:

- SAE J1939 for heavy-duty vehicles
- OBD2 for on-board diagnostics
- CANopen for embedded control applications such as industrial automation
- CAN FD an extension of classical protocol to flexible data-rate

To record CAN data, a CAN bus logger is used in order to temporally store the collected parameters on an SD card which manages data transmission to cloud servers. Then, raw data need to be decoded to transform them into a human-readable form. This operations depends on the structure of CAN bus data. Indeed, in order to decode raw CAN data, it is necessarily to know, for each CAN ID which parameters are included and, for each of them, the corresponding start bit and bit length. In passenger cars typically each manufacturer used its own protocol [7]. For what concerns heavy-duty manufacturers instead, SAE J1939 is a common adopted protocol that provides a set of standard messages and conversion rules shared for agricultural, military, mining and construction vehicles. However, there is also the possibility of collecting customized messages for which decoding extra information are required.

Since the vehicle made available for the analysis presented in the thesis is an heavy-duty one, the following considerations are based on the corresponding standard, SAE J1939.
It represents a growing in importance protocol, thanks to the wide spread of in-vehicle connectivity also for heavy-duty vehicles, supported by IoT solutions and the firmly established cars connected mobility.

A SAE J1939 message is composed by 93 bits: the first 29 represent the CAN identifier, while the last 64 consist in the data field. From the CAN identifier it is possible to obtain the Parameter Group Number (PGN), starting at bit 9 and with length 18. The first 9 bits correspond to the source address, while the last 3 define the message priority.
To each PGN corresponds different Suspect Parameter Numbers (SPN). Each SPN identifies completely the measured CAN parameter and it is also used to define the message priority, which is inversely proportional to the SPN value. For this reason, CAN messages characterized by SPN smaller than 30000 are standard, while the ones associated with higher SPNs are customized. The SPN is one of the information contained in the data field and gives the interpretation of the logged 8 bytes of raw data, representing the measured value for the parameter identified by the SPN.

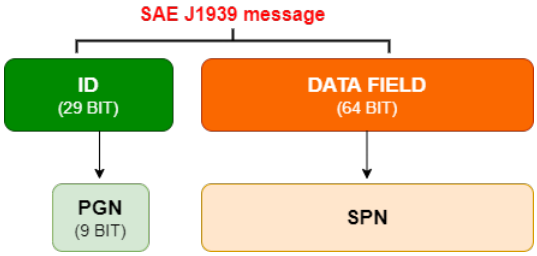

Figure 2.1: SAE J1939 Message Diagram

2.2 CAN bus data exploration

To validate the proposed procedures and analysis, they have been applied to a real use case by collecting data from a vehicle made available for tests and researches performed by Tierra. The vehicle under analysis is an almost fifteen years old farm tractor used to test both devices and driver assistance systems. This has two main implications that should be kept in mind to correctly interpret results. The former is that, since the vehicle is pretty old, it is not easy to decode CAN messages, especially customized ones: indeed, the standard might be changed or the additional extra information needed to decode them might not be updated or provided at all. The latter, since the vehicle is used to test assistance systems, data are not collected during actual working cycles: they could be, sometimes, simulated working phases whereas the behaviour during other periods could be pretty unusual for a heavy duty vehicle of the same type that is actually employed for working activities.

To collect data, an experimental *"multipurpose, remote configurable, secure, CAN bus data logger"* [1] has been installed on the vehicle.
Since the logger has AWS upload capabilities, raw data are stored in txt files that are sent to the Amazon storage service either when the vehicle is turned off or if the maximum memory size is reached (3.6 MB). Then, the Tierra parser script is applied to each txt file in order to decode its information. As previously introduced, since only CAN IDs with known number of parameters and structure can be decoded, for each input file two output txt files are created: one called *Parsed*, where CAN IDs associated with extra information found inside Tierra databases are decoded, and another one called *notFound* containing CAN IDs that can not be decoded because the required information are not known by Tierra. The only information available from the *notFound* files is the one related to the PGN. For *Parsed* files instead, the information decoded from raw data is combined with some metadata, so that each row of the resulting files is composed by 7 fields:

- Source address
- PGN
- Timestamp (in unix format)
- Message's description
- SPN

[1] www.tierratelematics.com

- Measured value for the considered SPN
- Unit of measurement

To avoid memory issues and high computational cost in the data import phase, only the information related to SPN, Timestamp and Measured value are considered. Indeed, any other additional information can be easily obtained in case of needs by querying Tierra databases.

The available final dataset consists then in 62,419,883 observations explained by 3 variables and the first rows are shown in Figure 2.2. As it is possible to see, each row refers to a single SPN measure, taken at the given and indicated timestamp.

	SPN	timestamp	measurement
0	30011	1573144720809	0.000
1	30011	1573144720859	0.000
2	32061	1573144720860	0.000
3	90	1573144720861	215.000
4	524	1573144720862	1.000
5	190	1573144720862	644.250
6	30789	1573144720864	191.000
7	190	1573144720886	643.375
8	30000	1573144720894	37.000
9	31800	1573144720905	16449.000
10	31391	1573144720907	0.000
11	190	1573144720908	643.750

Figure 2.2: First rows of dataset

The final dataset contains a set of parameters useful to describe the vehicle and the engine status, such as engine speed, percent load, coolant temperature, digging depth, etc.

All the observations come from 40 different days, from November 7, 2019 to April 15, 2020. However, the histogram plot reported in Figure 2.3 shows a quite irregular behaviour concerning the number of messages sent per day, highlighting the relative irregular and heterogeneous behaviour of the working time per day, typically associated with heavy-duty vehicles as the one under analysis [4]. Indeed this kind of vehicles, because of the specific characteristics of their working sites and the nature of the performed tasks, are often associated with a highly variable number of working hours per day that makes their usage pattern identification particularly complex.

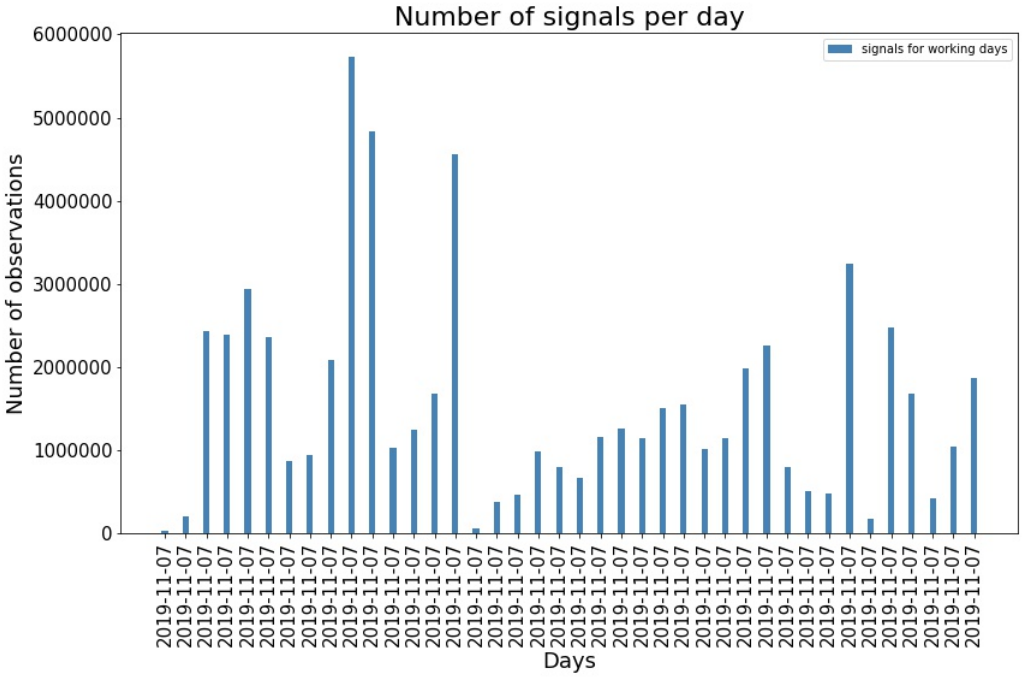

Figure 2.3: Number of observations per day

There are no null or missing values. However, the dataset contains some duplicate rows (about 0.08% of the entire dataset): because of the higher precision of timestamp with respect to the sampling rates, they are more likely to be due to problems connected with the network or the environment rather than two distinct observation sampled with time delta smaller than 1 millisecond. For this reasons, they are removed.
Furthermore, it happened that for a given SPN and timestamp, two different measures are collected. After checking that all the values collected in the given timestamps are feasible (according to the full scales values for the given SPN) and that they are relative close in magnitude, the same reasoning for solving the previous issue are applied and only one observation for each pair timestamp and SPN is considered for further analysis.

The dataset contains 20 different SPNs. It seems to be a limited number with respect to the expected one. Possible explanations could be either that the analyzed vehicle is quite old or it can be due to the fact that the considered model is not registered in Tierra database: hence, some SPN (especially customized ones) are not recognized by Tierra parser and can not be decoded, providing just the information about their PNG.

Since from a preliminary exploratory analysis carried out considering just the

first 40 files, a one-to-one relationship between SPNs and PGNs was noticed, the number of distinct total (*Parsed* + *notFound*) PGNs can be used as a rough approximation of the total number of distinct SPNs. Hence, it is possible to say that the number of distinct monitored parameters is 40, but only 20 of them have the additional information that make them in a suitable format for future analysis.

The considered 20 SPNs, their description, feasible range and unit of measurement are summarized in Table 2.1. Observations out of feasible ranges are removed since probably due to errors in measurement or during transmission or decoding phases.

SPN	Description	Feasible range
81	Engine diesel particulate filter inlet pressure	0 to 125 kPa
90	Power takeoff oil temperature	-40 to 210 deg C
94	Engine fuel delivery pressure	0 to 1000 kPa
110	Engine coolant temperature	-40 to 210 deg C
114	Net battery current	-125 to 125 A
123	Clutch pressure	0 to 4000 kPa
164	Engine injection control pressure	0 to 251 Mpa
182	Engine trip fuel	0 to 2105540607,5 L
183	Engine fuel rate	0 to 3212.75 L/h
190	Engine speed	0 to 8,031.875 rpm
524	Transmission selected gear	-125 to +125
975	Estimated percent fan speed	0 to 100 %
1638	Hydraulic temperature	-40 to 210 deg C
30000	Engine percent load	0 to 250 %
30011	Front plow swith	-
30694	Rear hitch position	-
30789	Charge pressure	-
31391	Amount of particulate matter C method	-
31800	Digging depth	-
32061	Fuel Tank Level	-

Table 2.1: SPNs: description and feasible range

In Figure 2.4 it is possible to see that the behaviour of SPNs is quite different: indeed, some of them are really frequent while others are characterized by a lower average sampling rate. Indeed CAN messages are transmitted at different rates

because of the specific characteristics of the CAN bus data [8]. Additionally, these rates are not constant, hence observations are not equally spaced in time. In Chapter 3 some strategies applied to overcome both different rates and not constant rates are presented. Having evenly time-spaced and synchronized observations is fundamental in order to apply some specific algorithms. Indeed, obtaining a constant sampling rate for each SPN allows to consider the given signal as a evenly spaced univariate time series whereas obtaining signals sampled at the same instants of time allows to consider the set of SPNs measurements as a multivariate time series.

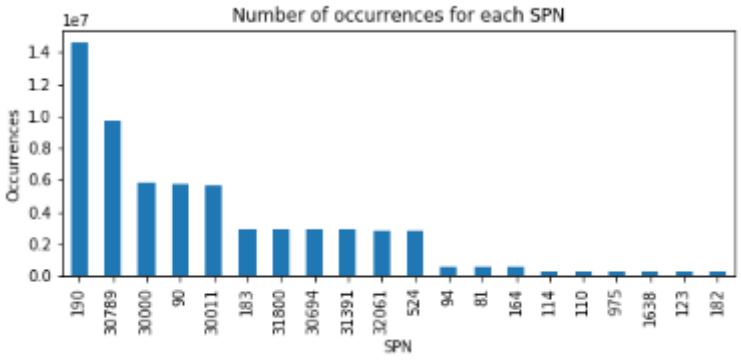

Figure 2.4: SPNs distributions

From the results of the analysis of the time delta distribution between consecutive observations of the same SPN, a distinction among SPNs can be made. More in details, four different groups can be identified: the first one is composed by regular and frequently sampled SPNs, such as the engine speed (SPN 190), charge pressure (SPN 30789) and engine load (SPN 30000). The second one contains parameters regularly sampled, but less frequently with respect to the previous group. It is the case, for example, of the engine fuel rate (SPN 183), rear hitch position (SPN 30694), amount of particulate matter (SPN 31391) and transmission selected gear (SPN 524). The third group, characterized by regularly and more rarely sampled SPNs, contains the engine coolant temperature (SPN 110), the estimated percent fan speed (SPN 975) and the engine fuel delivery pressure (SPN 94). Finally, the last group contains the front plow swith (DIG0) (SPN 30011) and the power takeoff oil temperature (SPN 90), both characterized by a high number of occurrences, but their distribution is not centered on a single value, meaning that they are collected at quite irregular timestamps. Such a behaviour could be due, for example, to device failures or they could be malfunction tell-tales. In any case, they should be treated in a different way from other SPNs.

Indeed, for centered distributions, it is possible to approximate the sampling rate with the inverse of 50th percentiles of time delta distribution (that coincides with its

mode) since the time delta are mostly centered on this value. This approximation results quite wrong instead for SPN 30011 and 90, for which another technique for estimating their sampling rate should be applied. However, since they are constant all over the available observations, they are not considered in further analysis, as well as regularly sampled constant SPNs.

The approximated time delta between consecutive observations of the 13 not constant SPNs are summarized in table 2.2, while the distribution of a representative SPN for each group is shown in Figure 2.5.

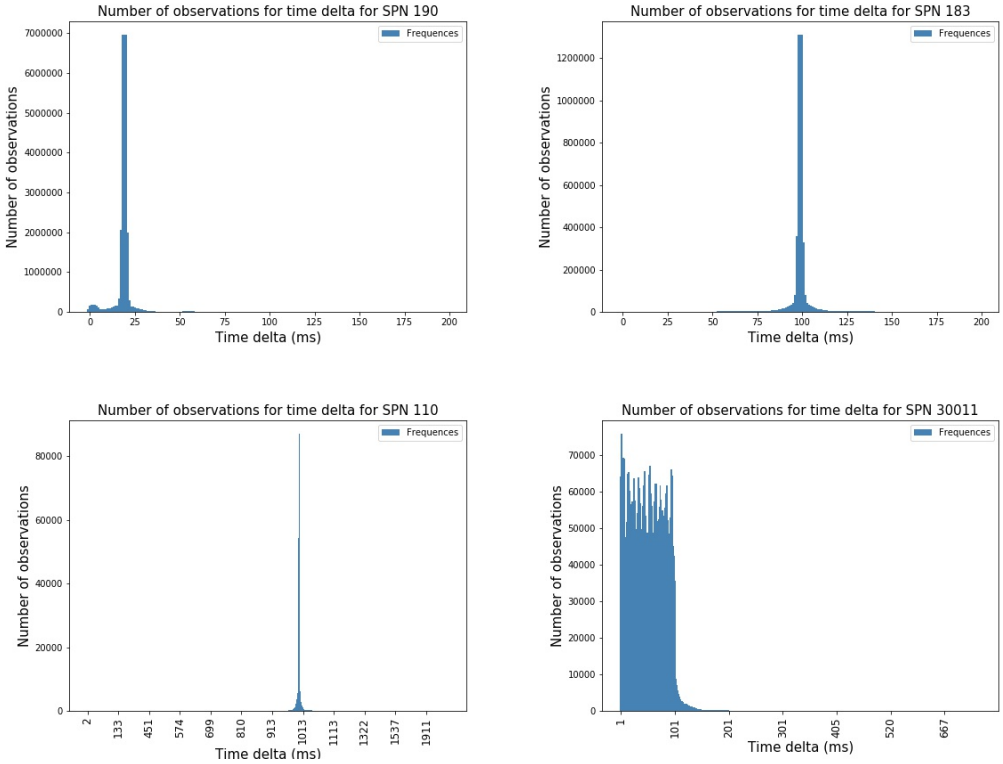

Figure 2.5: Time delta histograms for representative SPNs, namely SPN 190, 183, 110 and 30011

For display purpose, only shorter time deltas are shown in figure. However, data contains also observations spaced by higher temporal lags, associated with few occurrences, that are deeply analysed in the following since they could indicate shut downs or some kind of malfunctions.

SPN	Approximated time delta (s)
94	0,5
110	1
182	1
183	0,1
190	0,02
524	0,104
975	1
30000	0,05
30694	0,1
30789	0,03
31391	0,1
31800	0,1
32061	0,104

Table 2.2: SPNs approximated time delta between consecutive observation

Thanks to the obtained approximated values, it is possible to determine the timestamps corresponding to the moments at which the vehicle is switched off and, consequentially, the working cycles duration.
Indeed, the available data are collected only when the vehicle is on and therefore a sufficiently long time delta between consecutive observations suggests that the vehicle was turned off in the meanwhile. However, it is important to correctly quantify a *sufficiently* long time delta in order to tell missing data or transmission errors from the actual shutdowns.
Hence, to detect shutdown timestamps, the engine coolant temperature (SPN 110) was used: indeed, it represents a quite regularly sampled signal, so that a time delta higher than its average value, combined with a low measurement, suggests that the vehicle was actually turned off for a sufficiently long time such that the engine coolant reaches lower temperatures. Also time deltas higher than its 99.956th percentiles without a corresponding reduction in the engine coolant temperature is considered a shutdown: indeed, as it is possible to see from Figure 2.6, there are two specific days in which there are really short working cycles, probably due to tests on driver assistance studies carried out on the same vehicle, performed with smaller interruptions in the middle such that the engine coolant had no enough time to cool down. However, while the first method ensures that the identified timestamps actually correspond to shutdowns, using the latter method extra information are required to tell missing values and error transmissions from actual shutdowns.

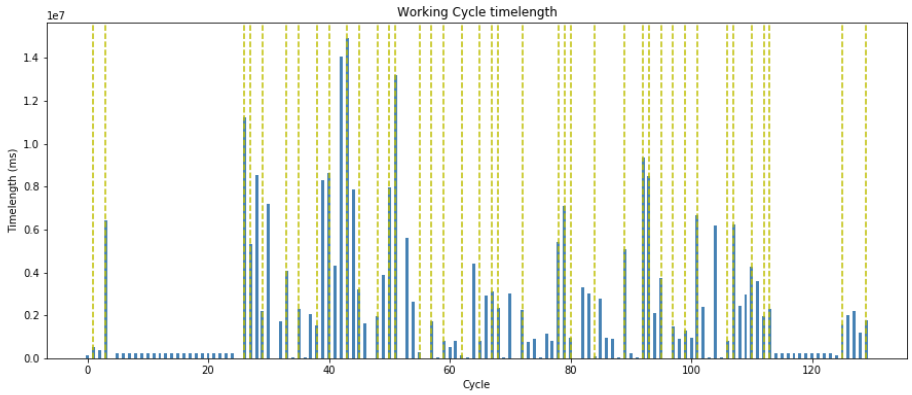

Figure 2.6: Working cycles length
The yellow vertical lines identify the different working days

Following this procedure, 130 different working cycles are identified. However, as it is also possible to see from Figure 2.6, from a deeper analysis it appears that five of them were extremely short, such that some SPN were not measured at all. Since they are in a limited number, probably not enough informative for further analysis and most likely due to transmission errors rather than actual working cycles, they are removed and not considered in the following.

As a conclusion of the exploratory data analysis, it is possible to visualize some of the considered SPNs: the first one corresponds to the engine coolant temperature, the parameter chosen to identify working cycles. As it is possible to see, most of working cycle starts correspond to it lower values, probably denoting the vehicle ignition. The second shown parameter is the engine speed, particularly relevant for the following analysis. The third SPN, the transmission selected gear, differs from all others because it is a categorical variable. Finally, the last SPN describes the engine trip fuel, an incremental value with a trivial upward trend. It is characterized by a regularly increasing behaviour in all points, with the exception of a single point of discontinuity: as a result of a deeper analysis it is possible to conclude that in the middle of these observations there is a lag of about two months in which no data are collected by the device. Such a behaviour can be used to identify device or transmission failures.

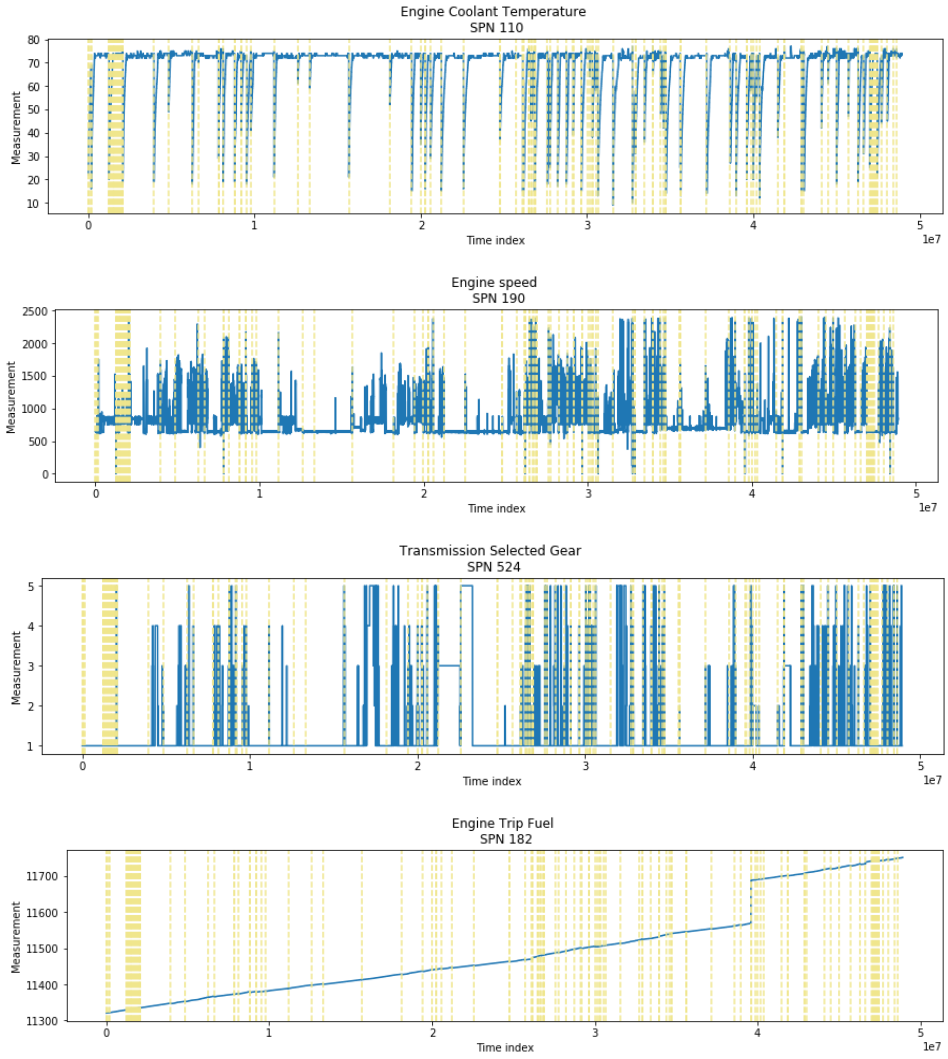

Figure 2.7: Plot over time for SPN 110, 190, 524 and 182
The yellow vertical lines identify the different working cycles

Finally, the distribution of the quantitative SPNs can be visualized and summarized thanks to boxplot. More in detail, for the 12 selected quantitative variables, minimum, first quartile, median, third qualite and maximum are graphically displayed. Indeed, the box goes from first to third quartile, while the horizontal line inside the box denotes the median. The whiskers are instead used to describe the data range. Using the provided function in Pandas [2] library, the whiskers are set

[2]pandas.pydata

by default to a distances from box borders equal to 1.5 times the interquartile range, defined as the difference between the third and the first quartile. Points outside the range given by whiskers are commonly considered outliers. In many applications, outlier removal is one of the main steps of data cleaning. However in this application they are taken into account, since observations outside full scale values have been already removed and extreme values could be useful to identify common patterns and make a distinction among different vehicle usages.

Boxplots are a descriptive non-parametric tool used to highlight the dispersion and the skewness of the variables under analysis by means of the distances between the five previously described quantity used to summarize the data distribution. As it is possible to see from Figure 2.8, some distributions seem to be quite asymmetric since they are more skewed to extremer values: it is the case of the engine speed, the engine percent load and the engine fuel delivery pressure. On the other hand, some SPNs are instead quite symmetric, such as the engine trip fuel. In addition, it is possible to notice that some SPNs are characterized by a higher dispersion, in the sense that they assume frequently a wider range of values, such as the engine trip fuel, the engine percent load and the engine speed, while others distributions are more flatted on few values, such as the digging depth, the estimed percent fan speed and the amount of particulate C matter method.

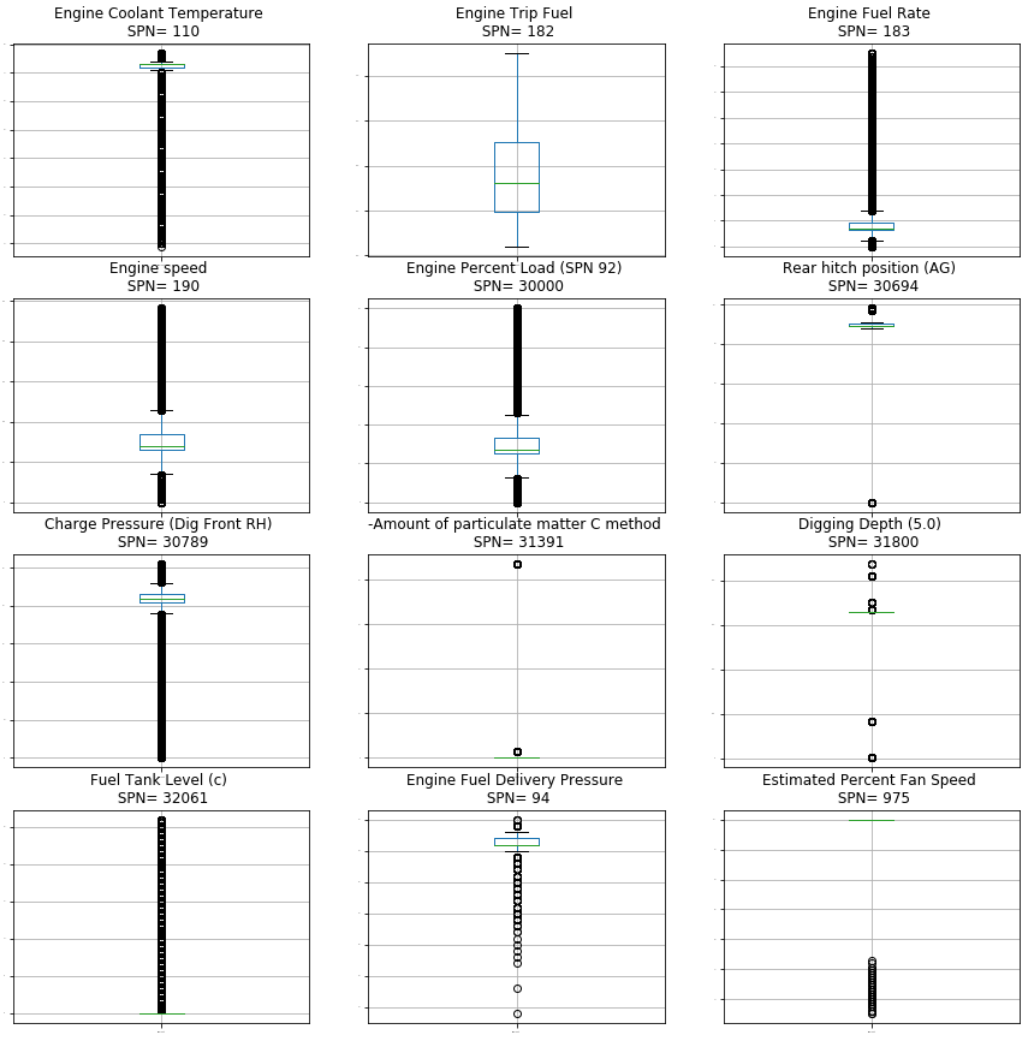

Figure 2.8: Boxplot for all the quantitative variables in the dataset, namely SPN 110, 182, 183, 190, 30000, 30694, 30789, 31391, 31800, 32061, 94, 975.

Chapter 3

CAN bus signal analysis and alignment

In the previous chapter, the two main issues related to the sampling of signals have been highlighted. Namely, CAN messages are transmitted at different rates because of the specific characteristics of the CAN bus data [8]. Additionally, these rates are not constant, hence observations are not equally spaced in time. Hence, in this chapter signals alignment is performed. Indeed, evenly time-spaced observations is fundamental for further analysis since several algorithms and methods require a constant sampling rate. For these reasons, signals alignment is considered one of the fundamental preprocessing steps for multivariate time series analysis [9]. Furthermore, since the aim is to describe the vehicle state using the entire set of available SPNs, considering each of them as a component of a multivariate time series, observations need to be taken at the same instants of time.

In order to obtain aligned and evenly spaced data points, a technique based on signal processing analysis and Fourier transform is applied. It is based on performing a downsampling operation in order to obtain all signals sampled at the lowest sampling rate. This procedure will cause a loss of information, but on the other hand it is preferable over oversampling since it will not introduce extra features in the original data [7]. In addition, since according to domain experts this type of data is quite noisy, downsampling in frequency domain is more appropriate. The choice of downsampling to the lowest sampling rate by frequency domain interpolation is supported and commonly adopted in CAN data analysis [8]. Indeed, since SPNs measure physical quantities, a higher rate would introduce redundancy in data. Besides the computational advantages, reducing data granularity will improve also costs due to data transmission on the network.

Since SPNs measurements can be seen as digital signals, the main signal processing principles are presented in the following in order to introduce the tools

used to synchronize time series, such as the discrete-time Fourier transform (and the Fast Fourier Transform algorithm), filters and upsampling and downsampling systems. In the second part of the chapter instead it is described how these tools have been applied to the real case scenario under analysis.

3.1 Signal processing fundamentals

A *signal* is a function containing the information about a phenomenon over time. It is said to be an *analog* signal if it is real valued and defined at each point in time. In contrast, it is said to be *digital* if it is discrete-time and takes values on a discrete set.
There exist also some intermediate situations: a signal is said to be a *samples sequence* if it is discrete-time but real valued, while it is *quantized* if discrete valued and defined at each point in time.

One or more signals can be the inputs of a *system* that can be seen as a device that, given one or more inputs, will produce one or more signals as output, also called *responses*.
The relationship between input and output signals can be of several forms and systems can be characterized by the properties of the operator identifying the system itself.

From now on, digital signals are considered because of the following applications in the thesis work.

A digital signal is a sequence in the vector space $H = \mathbf{C}^{\mathbf{K}}$, where $\mathbf{K}$ can be a finite subset of $\mathbf{Z}$ or coincides with $\mathbf{Z}$ itself.
In the former case, the digital signal is finite length. It can be extended to an infinite length signal using different methods. Examples can be zero padding, symmetric extension consisting in left-flipping the signal at the beginning and right-flipping it at the end, or periodic extension consisting in replicating the signal at the beginning and at the end.

A given signal can be represented using different vector bases [10]:

Definition 3.1.1 *Given a normed vector space V, a set of vectors $\{\varphi_k\}_{k\in\mathbf{K}}$, with K finite or countably infinite, is called a* basis *for V if the following conditions are satisfied:*

- *it is a* complete set*:*

$$\forall x \ \in V \ \exists \alpha \in \mathbf{C}^{\mathbf{K}} \text{ such that } \mathrm{x} = \sum_{\mathrm{k}\in\mathbf{K}} \alpha_{\mathrm{k}} \varphi_{\mathrm{k}}$$

- $\alpha \in \mathbf{C^K}$ *is unique*

Hence, according to definition (3.1.1), given a signal x and fixed a basis $\{\varphi_k\}_{k\in\mathbf{K}}$, the signal expansion with respect to the chosen basis is given by a linear combination of these functions with suitable coefficients:

$$x = \sum_{k\in\mathbf{K}} \alpha_k \varphi_k \tag{3.1}$$

There exist different basis choices that can be made, each with its own proprieties that can highlight different features of the same original signal [11], [10]. Usually, useful proprieties that a basis should satisfied are:

- *Sparsity*: few coefficients α_k are different from zero or significant.
- *Interpretability*: some features of the original signal can be directly extracted from coefficients α_k.

The category of bases commonly considered is *orthonormal bases* [10]:

Definition 3.1.2 *Given an Hilbert space H, a set of vectors $\{\varphi_k\}_{k\in\mathbf{K}}$, with $\mathbf{K}$ finite or countably infinite, is said to be an* orthonormal basis *for H if the following conditions are satisfied:*

- *it is a basis for H*
- *it is* orthonormal*:*

$$< \varphi_i, \varphi_j >= \begin{cases} 1 & \text{if i = j} \\ 0 & \text{if i} \neq \text{j} \end{cases}$$

A sequence $\{\varphi_k\}_{k\in\mathbf{J}}$ that satisfied only the latter condition is said to be an orthonormal sequence*.*

In case of an orthonormal basis,

$$H = \overline{\text{span}\,\{\varphi_\text{k}\}_{\text{k}\in\mathbf{K}}}$$

and the coefficients in (3.1) can be expressed as the inner product of the considered Hilbert space,

$$\alpha_k =< x, \varphi_k >$$

thanks to the *Riesz representation theorem*:

Theorem 1 *Given an Hilbert space H and an orthonormal basis $\{\varphi_k\}_{k\in\mathbf{K}}$ of H, then the application*

$$\begin{aligned}\phi^* : \quad H &\longrightarrow l^2(\mathbf{K}) \\ x &\longrightarrow \{< x, \varphi_k >\}_{k\in\mathbf{K}}\end{aligned}$$

is an isometric isomorphism.

The operator ϕ^* is called *analysis operator*. Hence, the Parseval's identity holds

$$||x||_H^2 = \sum_{k\in\mathbf{K}} | < x, \varphi_k > |^2$$

and the inverse application ϕ, also called *synthesis operator*, is defined as

$$\begin{aligned}\phi : \qquad l^2(\mathbf{K}) &\longrightarrow H \\ \alpha = \{\alpha_k\}_{k\in\mathbf{K}} &\longrightarrow \sum_{k\in\mathbf{K}} \alpha_k\varphi_k\end{aligned}$$

so that $\forall x \in H$

$$x = \sum_{k\in\mathbf{K}} < x, \varphi_k > \varphi_k$$

From previous equations it is clear that the synthesis operator is the adjoint of the analysis operator [10].
By definitions, in the case of an orthonormal basis,

$$\phi^*\phi = \phi\phi^* = I$$

where I denotes the identity operator. Hence ϕ is *unitary*.

Examples of commonly used orthonormal bases for $\mathbf{C}^K$are *Fourier* or *Wavelet* basis, definided as it follows:

Definition 3.1.3 *The* Fourier basis *for the space $\mathbf{C}^K$ is defined by the set of vectors $\left(\varphi^0, \ldots, \varphi^{K-1}\right)$ where*

$$\varphi_n^k = \frac{1}{\sqrt{K}} e^{\frac{2\pi}{N} ikn}, \qquad k = 0, \ldots, K-1,\ n = 0, \ldots, N-1$$

The bounds on n and k are due to the fact that the considered functions are N periodic thanks to the complex exponential property $e^{li2\pi} = 1 \quad \forall l \in \mathbf{Z}$.

The angular frequency of φ^k is given by $\omega_k = \dfrac{2\pi}{N}k$ while the frequency can be obtained as $f_k = \dfrac{k}{N}$.
Hence, the maximum angular frequency is $\omega_{\max} = \pi$.

The defined Fourier basis is orthonormal:

$$\begin{aligned} < \varphi^k, \varphi^m > &= \frac{1}{N} \sum_{n=0}^{N-1} e^{\frac{2\pi}{N} ikn} e^{\frac{-2\pi}{N} imn} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} e^{\frac{2\pi}{N} i(k-m)} \\ &= \frac{1 - e^{\frac{2\pi}{N}(k-m)N}}{1 - e^{\frac{2\pi}{N} i(k-m)}} \\ &= \begin{cases} 1 & \text{if k = m} \\ 0 & \text{if k} \neq \text{m} \end{cases} \end{aligned}$$

Definition 3.1.4 Wavelets *are defined as a families of orthogonal basis functions obtained from a mother wavelet* ψ *by translation and dilation as*

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \psi \left(\frac{t - 2^j k}{2^j} \right)$$

and from a father wavelet ϕ *by dilation and translation as*

$$\phi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \phi \left(\frac{t - 2^j k}{2^j} \right)$$

with $j = 1,2,\ldots$ *is the index referred to the scale and* $k = 1, \ldots, \frac{N}{2^j}$ *indexes the translation location in time [12].*

Wavelet decomposition is often used to exploit its *whitening* or *decorrelating* property: indeed, writing the signal with respect to a Wavelet basis represents a way to deal with highly autocorrelated signals since what is typically observed is that Wavelet coefficients are characterized by a smaller and often negligible correlation [13].

If instead $\{\varphi_k\}_{k \in \mathbf{J}}$ is an orthonormal sequence, then it is a basis for the vector subspace $S = \overline{\text{span}\,\{\varphi_\text{k}\}_{\text{k} \in \mathbf{J}}} \subset H$. In this case is still possible to define the analysis and synthesis operator in an analogous way as previous case:

$$\begin{aligned} \phi^* : H &\longrightarrow l^2(\mathbf{J}) \\ x &\longrightarrow \{< x, \varphi_k >\}_{k \in \mathbf{J}} \end{aligned} \tag{3.2}$$

$$\begin{aligned} \phi : \ l^2(\mathbf{J}) &\longrightarrow H \\ \alpha = \{< x, \varphi_k >\}_{k \in \mathbf{J}} &\longrightarrow \hat{x} = \sum_{k \in \mathbf{J}} \alpha_k \varphi_k \end{aligned} \tag{3.3}$$

The property $\phi^*\phi = I$ still holds but

$$\phi\phi^* = P_S$$

where P_S denotes the orthogonal projection onto the subspace S.

$\hat{x}$ can be interpreted as an approximation of x. It is called *linear approximation* if the projection is onto a fixed M dimensional subspace (hence independent on the signal x), *non-linear approximation* if the projection is onto the M dimensional subspace generated by the M most important basis vectors with respect to x variations. A way of characterizing the importance of these vectors is considering the largest inner products with x.

The error of the approximation can be computed as

$$\varepsilon = ||x - \hat{x}||^2 = ||\tilde{x}||^2$$

where $\tilde{x}$ is the projection of x onto the orthogonal complement of S.

Studying the behaviour of the approximation error using different basis decompositions, is possible to note that different bases yield to different results. Indeed, it is possible to prove [10] that for linear approximation $\varepsilon \sim \frac{1}{M}$ both for Fourier and Wavelet basis. However, in the case of non linear approximation, $\varepsilon \sim \frac{1}{M}$ for Fourier decomposition while $\varepsilon \sim \frac{1}{2^M}$ for Wavelet one [14], [10].

3.1.1 Discrete-time systems

As previously introduced, a system is an operator that, given one or more input signals, will produce one or more signals as output. A system is said to be *discrete-time* if the input signal is a sequence that is mapped by the system in another output sequence.

Given an input sequence x in a space V, the output of the system A can be written as $y = Ax$.

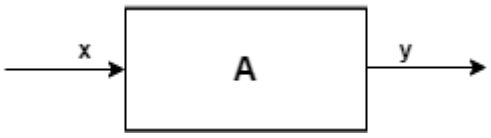

Figure 3.1: System

Definition 3.1.5 *A discrete time system is said to be* linear *if A is linear operator:* $\forall x_1, \ldots, x_n$ *input signals,* $\forall \alpha_1, \ldots, \alpha_n \in \mathbf{C}^n$

$$A\left(\sum_{i=1}^{n} \alpha_i x_i\right) = \sum_{i=1}^{n} \alpha_i A\left(x_i\right) = \sum_{i=1}^{n} \alpha_i y_i \tag{3.4}$$

Property (3.4) is also known as *superposition principle.*

An example of linear system is the one described by the *shift* operator $T_k,\ k \in \mathbf{Z}$, whose action can be written as

$$y_n = T_k x_n = x_{n-k}$$

Linear operators can be naturally written in matrix form. This is particularly useful when the matrix structure reflects some properties of the linear system. An example, is given by *memoryless* property:

Definition 3.1.6 *A linear system is said to be* memoryless *if the matrix A is diagonal.*

Previous definition means that the n-th component of output signal depends only on the n-th component of the input one.

Definition 3.1.7 *A discrete-time system is said to be* shift-invariant *if*

$$AT_k x = T_k A x \quad \forall k \in \mathbf{Z}$$

where T_k denotes the shift operator.

Definition 3.1.8 *A discrete-time system is called* BIBO *(bounded-input bounded-output)* stable *when a bounded input x produces a bounded output $y = Ax$.*

The previous definition can be rewritten as

$$A \text{ BIBO} \iff A : l^\infty(\mathbf{Z}) \longrightarrow l^\infty(\mathbf{Z})$$

where $l^\infty(\mathbf{Z}) = \left\{ x = \{x_k\}_{k\in\mathbf{Z}} \ : \ ||x||_\infty = \sup_{k\in\mathbf{Z}} |x_k| < \infty \right\}$

Linear time-invariant stable operator $H : l^2(\mathbf{Z}) \longrightarrow l^2(\mathbf{Z})$ represent a relevant class of systems, also called *filters.*

Their importance is due to the fact that can be completely described by their *impulse response* [11]:

Definition 3.1.9 *Given a linear time-invariant stable operator $H : l^2(\mathbf{Z}) \to l^2(\mathbf{Z})$, the* impulse response *h is defined as the output produced by system H when it takes as input the Kronecker delta sequence δ:*

$$h = H\delta$$

$$\text{with } \delta_k = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}$$

The Kronecker delta sequence δ can be used to express the canonical basis of $l^2(\mathbf{Z})$ by means of the shift operator as

$$e_k = T_k\delta, \quad k \in \mathbf{Z}$$

Then, knowing the impulse response h, the response variable of system H when it takes as input $x = \sum_{k\in\mathbf{Z}} x_k T_k\delta$ can be written, thanks to H linearity, stability, shift-invariance and the given definition of impulse response respectively, as

$$\begin{aligned} Hx &= \sum_{k\in\mathbf{Z}} x_k T_k\delta = \sum_{k\in\mathbf{Z}} x_k H T_k\delta \\ &= \sum_{k\in\mathbf{Z}} x_k T_k H\delta = \sum_{k\in\mathbf{Z}} x_k T_k h \end{aligned}$$

Hence, $y_n = (Hx)_n = \sum_{k\in\mathbf{Z}} x_k h_{n-k}$.

More formally, it is possible to define the output of a linear time-invariant stable system H from the definition of *convolution* [10]:

Definition 3.1.10 *Given two sequences x and h, their* convolution *is defined as*

$$(Hx)_n = (x * h)_n = \sum_{k\in\mathbf{Z}} x_k h_{n-k} = \sum_{k\in\mathbf{Z}} h_k x_{n-k}$$

H is hence called convolution operator*.*

Definition 3.1.11 *A filter is said to be [11]:*

- causal *if $h_n = 0\ \forall n < 0$,* anticausal *if $h_n = 0\ \forall n > 0$,* not causal *otherwise.*
- stable *if $h \in l^1(\mathbf{Z})$ or, equivalently, if $H : l^\infty \longrightarrow l^\infty$ is BIBO.*
- Finite impulse response *if $h_n \neq 0$ for a finite number of indices n,* infinite impulse response *otherwise.*

3.1.2 Fourier Transform

As shown in previous sections, in order to compute the response signal of a linear time-invariant bounded system it is necessary to compute a convolution. However, introducing an appropriate transformation, it is possible to rewrite the signal in a different space such that the convolution becomes simply a multiplication. It is sufficient to go from the time to the frequency domain, applying the so called *Fourier transform*.
In addition to computational advantages, it can be useful also to highlight, in the frequency domain, properties that were not inspectable in the time domain. Furthermore, since the Fourier basis has the *interpretability* property, it can be also used to approximate the signal considering only the components corresponding to the most important frequencies.

In particular, in the following the discrete-time Fourier transform (DTFT) is described: it is the Fourier transform version for infinite-length discrete-time signals x [10], while usually the term Fourier transform refers to continuous-time signals

Let H be a linear shift-invariant system and consider the sequence $v = \{v\}_{n\in\mathbf{Z}}$ composed by the complex exponential $v_n = e^{i\omega n},\ n \in \mathbf{Z}$.
$\omega \in \mathbf{R}$ is called *angular frequency*. It can be also written as $\omega = 2\pi f$, where f measures the number of cycles in a unit of time, also called *frequency*.

Since $|v_n| = 1\ \forall n$, $v \in l^\infty$. If $h \in l^1$, then $h * v$ is bounded and its components can be written as

$$\begin{aligned}(Hv)_n &= (h*v)_n = \sum_{k\in\mathbf{Z}} v_{n-k}h_k = \\ &= \sum_{k\in\mathbf{Z}} e^{i\omega(n-k)}h_k = \sum_{k\in\mathbf{Z}} h_k e^{-i\omega k}v_n\end{aligned}$$

Hence v is an eigensequence for the operator H.

The previous expression can be then rewritten as

$$Hv = H(e^{i\omega})v \tag{3.5}$$

where $H(e^{i\omega})$ is called *frequency response* or *transfer function*.

The discrete-time Fourier transform is then obtained projecting the signal onto the subspaces generated by each eigensequence [10]:

Definition 3.1.12 *Given a infinite-length discrete-time sequence x, its Fourier transform is defined as*

$$X(e^{i\omega}) = \sum_{n\in\mathbf{Z}} x_n e^{-i\omega n}, \qquad \omega \in \mathbf{R} \tag{3.6}$$

The previous expression shows that the discrete-time Fourier transform is a 2π periodic function of the angular frequency ω.
It is well defined if the expression (3.6) converges $\forall\omega \in \mathbf{R}$.

For example, if $x \in l^1(\mathbf{Z})$ then (3.6) is uniformly absolutely-convergent on R and $X(e^{i\omega})$ is a continuous function.

For series $x \notin l^1(\mathbf{Z})$, the convergence of (3.6) is not ensured. However, the discrete-time Fourier transform can be extended to $x \in l^2(\mathbf{Z})$ considering, instead of uniform convergence, the convergence in $L^2([-\pi, \pi))$ norm, that is equivalent to require that (3.6) converges for almost every $\omega \in \mathbf{R}$.

Indeed, considering the trigonometric polynomials defined as

$$X_N(e^{i\omega}) = \sum_{k=-N}^{N} x_k e^{-i\omega k}, \qquad N \in \mathbf{N}$$

the discrete-time Fourier transform is defined as

$$X(e^{i\omega}) = \lim_{N \longrightarrow +\infty} X_N(e^{i\omega})$$

if this limit exists.

Hence, the discrete-time Fourier transform can be defined as an isomorphism $\mathcal{F}$ that, given a sequence in $l^2(\mathbf{Z})$, maps it into a 2π-periodic function in the space $L^2(-\pi, \pi)$.
The inverse discrete-time Fourier transform (IDTFT) can be defined as [10]:

Definition 3.1.13 *Given a 2π periodic function $X(e^{i\omega})$, its inverse discrete time Fourier transform is defined as*

$$x_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{i\omega}) e^{i\omega n} d\omega \qquad n \in \mathbf{Z}$$

It is possible to prove that the *Parseval equality* for discrete-time Fourier transform holds:

$$||x||_{l^2}^2 = \int_{-\pi}^{\pi} |X(e^{i\omega})| \frac{d\omega}{2\pi}$$

3.1.3 Discrete Fourier Transform

The Discrete Fourier Transform can be seen as the sampled version of the Discrete-Time Fourier Transform. It is introduced to overcome some practical issues related to the discrete-time version. First of all, the discrete-time Fourier transform can not be handled by a computer because it is composed by an infinite number of values. Moreover, the discrete Fourier transform became popular because of the *Fast Fourier Transform* algorithm that allows to compute the discrete Fourier transform of a signal with $\mathcal{O}(N\log(N))$ operations instead of $\mathcal{O}(N^2)$, introducing computational advantages for obtaining the response signal of given a system.
Hence it can be thought as an operational tool for computing the Discrete-time Fourier Transform both for infinite-length periodic sequences both for finite-length ones, treated as if they arise from one period of an infinite-length periodic sequence. Indeed, they coincides if a finite segment of an infinite-length signal is considered.

Given a finite-length sequence $x \in \mathbf{C}^N$ and the Fourier basis of $\mathbf{C}^N$ defined in (3.1.3), using the previous introduced analysis operator (3.2) it is possible to write the finite-length signal as

$$x = \sum_{k=-N}^{N} \alpha_k \varphi^k$$

where α_k is the Fourier coefficient, given by

$$\alpha_k = < x, \varphi^k >$$

The set of coefficients with respect to the Fourier basis is called *spettrum* of x. The modulus of Fourier coefficients gives a description of how the signal is distributed over its discrete frequencies.

Furthermore, the Fourier coefficients of a real valued signal satisfy

$$|\alpha_{-k}| = |\alpha_k| \quad \forall k = -N, \ldots, N.$$

This means that in the case of real valued signal the spectrum is symmetric with respect to $\frac{N}{2}$ and negative coefficients add no information in spectrum analysis.

Neglecting the normalization factors $\dfrac{1}{\sqrt{N}}$ in the expression of the basis vectors, it is possible to show the relationship between the Discrete-time Fourier Transform and the Discrete Fourier Transform. Indeed, if the DTFT is sampled at points $\omega_k = \dfrac{2\pi}{N} k \; k \in \{0, 1, \ldots, N-1\}$, the expression of DFT is obtained [10]:

$$\begin{aligned} X(e^{i\omega})|_{\omega=\omega_k} &= X(e^{i\frac{2\pi}{N}k}) = \sum_{n\in\mathbf{Z}} x_n e^{i\frac{2\pi}{N}kn} = \\ &= \sum_{n=0}^{N-1} x_n e^{i\frac{2\pi}{N}kn} \end{aligned}$$

3.1.4 Multirate Systems

Two sequences are said to be *multirate* if each time index refers to a different time scale. To handle these situations, multirate system are introduced: they are the result of suitable combinations of filters and upsampling and downsampling operators.

Given a positive integer N, a sequence x is *downsampled* by N is

$$y_n = x_{Nn}$$

Its Discrete-Time Fourier Transform is given by

$$Y(e^{i\omega}) = \frac{1}{N}\sum_{k=0}^{N-1} e^{i\frac{\omega+2\pi k}{N}}$$

The corresponding operator is denoted by D_N and defined as it follows

$$\begin{aligned} D_N : \quad l^2(\mathbf{Z}) &\longrightarrow l^2(\mathbf{Z}) \\ x &\longrightarrow y = D_N x \end{aligned}$$

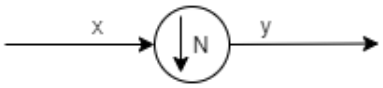

Figure 3.2: Block diagram for downsampling

Since downsampling contracts time, it will expand frequency accordingly.

The dual operation of downsampling is upsampling.
Given a positive integer N, a sequence x is *upsampled* by N is

$$y_n = \begin{cases} x_{n/N} & \text{if (n, N)} = 0 \\ 0 & \text{otherwise} \end{cases}$$

Its Discrete-Time Fourier Transform is given by

$$\begin{aligned} Y(e^{i\omega}) &= \sum_{n\in\mathbf{Z}} y_n e^{-i\omega n} = \sum_{n\ :(n,N)=0} x_{\frac{n}{N}} e^{-i\omega n} = \\ &= \sum_{k\in\mathbf{Z}} x_k e^{-Ni\omega k} = X(e^{iN\omega}) \end{aligned}$$

So the resulting spectrum is compressed.

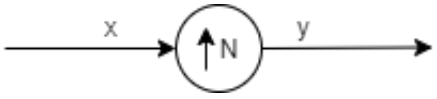

Figure 3.3: Block diagram for upsampling

The corresponding operator is denoted by U_N and defined as it follows

$$\begin{aligned} U_N: \quad l^2(\mathbf{Z}) &\longrightarrow l^2(\mathbf{Z}) \\ x &\longrightarrow y = U_N x \end{aligned}$$

Downsampling and upsampling operators can be combined: indeed, for any positive integer N

$$D_N U_N = I$$

while

$$P = U_N D_N$$

is the orthogonal projection operator over the l^2 vector subspace

$$\{\{x_n\} \ : \ x_n = 0 \text{ if } (\mathrm{n}, \mathrm{N}) \neq 0\}$$

In addition, it is possible to achieve any rational rate change by suitable combining upsampling by a factor M and downsampling by N. In this sense, they can be used in multirate sequences analysis for aligning time scale and, consequentially, scaling frequencies. However, the introduction of new frequencies in the spectrum of the original signal is often an undesired effect.
To avoid this behaviour, downsampling is often preceded by filtering and upsampling is often followed by filtering.

Indeed, considering $x_n \in l^2(\mathbf{Z})$, the *downsampling* system ϕ^* is composed by a filter characterized by impulse response g^*_{-n} followed by a downsampling by $N > 1$.

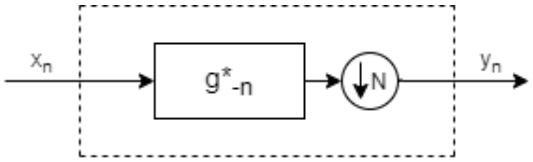

Figure 3.4: Block diagram for downsampling preceded by filtering

Given $x \in l^2(\mathbf{Z})$, the response variable of the sampling system can be obtained as it follows:

$$\begin{aligned} y_k &= (\phi^* x)_k = (g^*_{-n} * x_n)_{n=kN} = \\ &= \sum_{m \in \mathbf{Z}} x_m g^*_{m-kN} = < x, \varphi_k > \end{aligned}$$

where

$$\varphi_k = g_{n-kN} \; n \in \mathbf{Z} \tag{3.7}$$

In the following $\{\varphi_k\}_{k\in\mathbf{Z}}$ will be assumed to be an orthonormal set. This means that

$$< \varphi_k, \varphi_j >=< g_{n-kN}, g_{n-lN} >= \delta_{k-l} \quad \forall k, l \in \mathbf{Z}$$

Then y is the orthogonal projection of x onto the subspace S generated by the set $\{\varphi_k\}_{k\in\mathbf{Z}}$ while its projection onto $\overline{\text{span}\,\{\varphi_k\}_{k\in\mathbf{Z}}}$ represents the loss of information due to downsampling operations.
In this sense, any $x \in l^2(\mathbf{Z})$ has an orthogonal decomposition that can be written as $x = x_S + x_{S^\perp}$ where $x_{S^\perp}$ can not be reordered from y.

The *upsampling* system ϕ is composed instead by an upsampling by $N > 1$ followed by a filter characterized by impulse response g_n.

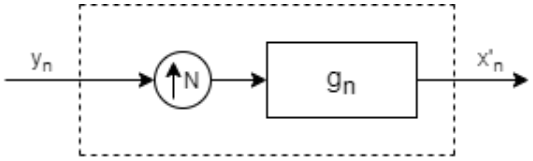

Figure 3.5: Block diagram for upsampling followed by filtering

Given any $y \in l^2(\mathbf{Z})$, the output of the upsampling system x' can be computed as

$$x'_n = (\phi y)_n = \sum_{k\in\mathbf{Z}} y_k g_{n-kN} = \left(\sum_{k\in\mathbf{Z}} y_k \varphi_k\right)_n$$

where φ_k is defined in (3.7).

Combining upsampling and downsampling operators, it is possible to prove that the following properties hold:

- $\phi^*\phi = I \iff \{g_{n-kN}\}_{k\in\mathbf{Z}}$ is an orthonormal set
- $\phi\phi^* = P$ where P denotes the orthogonal projection onto the subspace generated by $\{\varphi_k\}_{k\in\mathbf{Z}}$

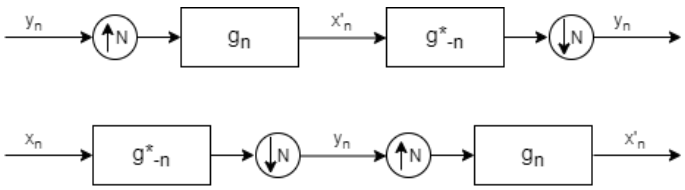

Figure 3.6: Block diagram for the combination of upsampling and downsampling operators

An important application of the previous described systems is to *bandlimited* signals:

Definition 3.1.14 *A signal $x \in l^2(\mathbf{Z})$ is said to be* bandlimited *if there exists $\omega_0 \in [0, 2\pi)$ such that the discrete-time Fourier transform X of x satisfied*

$$X(e^{i\omega}) = 0 \quad \forall\omega \ : \ |\omega| \in \left(\frac{\omega_0}{2}, \pi\right]$$

The smallest ω_0 is called bandwidth *of x.*
If such ω_0 does not exist, then x is a full-band *sequence.*

Indeed, it is possible to prove that the set of sequences in $l^2(\mathbf{Z})$ with bandwidth limited by ω_0 is a closed subspace, denoted by $BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$.

Hence, the previous described procedure for downsampling and upsampling can be applied to project an infinite-length signal x onto $BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$.

For $N > 1$, let

$$g_n = \frac{1}{\sqrt{N}}\text{sinc}\left(\frac{\pi \text{n}}{\text{N}}\right), \ \text{n} \in \mathbf{Z} \tag{3.8}$$

The corresponding Discrete-Time Fourier Transform is

$$G(e^{i\omega}) = \begin{cases} \sqrt{N} & \text{if } |\omega| \leq \frac{\pi}{\text{N}} \\ 0 & \text{otherwise} \end{cases}$$

Previous expression shows that g_n and its shift belong to $BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$.

Furthermore, because of the generalized Parseval equality, it can be proven that they are orthonormal:

$$\begin{aligned} < g_{n-kN}, g_{n-lN} > &= \frac{1}{2\pi} < e^{-i\omega kN} G(e^{i\omega}), e^{-i\omega lN} G(e^{i\omega}) >= \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\omega(k-l)N} |G(e^{i\omega}|^2 d\omega = \\ &= \frac{N}{2\pi} \int_{-\frac{\pi}{N}}^{\frac{\pi}{N}} e^{-i\omega(k-l)N} = \delta_{k-l} \end{aligned}$$

Hence for any positive integer N, combining ϕ and ϕ^* as previously described and considering as filter the one described by g_n as defined in (3.8), it is possible to prove [10] that

$$x'_n = \frac{1}{\sqrt{N}} \sum_{k \in \mathbf{Z}} y_k \text{sinc}\left(\frac{\pi}{\text{N}}(\text{n} - \text{kN})\right) \quad \text{n} \in \mathbf{Z}$$

with

$$y_k = \frac{1}{\sqrt{N}} \sum_{n \in \mathbf{Z}} x_n \text{sinc}\left(\frac{\pi}{\text{N}}(\text{n} - \text{kN})\right) \quad \text{k} \in \mathbf{Z}$$

is the best approximation of x in $BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$.

In the general case the effect of the orthogonal projection onto $BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$ to truncate the spectrum of x to $\left[-\frac{\pi}{N}, \frac{\pi}{N}\right]$ so that the Discrete-Time Fourier Transform of the resulting signal can be written as

$$\hat{X}(e^{i\omega}) = \begin{cases} X(e^{i\omega}) & \text{if } |\omega| \leq \frac{\pi}{\text{N}} \\ 0 & \text{otherwise} \end{cases}$$

If $x \in BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$ instead, then $x = x'$ so that it is possible to completely recover the original signal from its samples.
In this case, the effect of the prefilter is simply to scale the signal by $\sqrt{N}$ and the *sampling theorem for sequences holds* [10]:

Theorem 2 *Given a positive integer N, if $x \in BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$ then*

$$x_n = \sum_{k \in \mathbf{Z}} x_{kN} \text{ sinc}\left(\frac{\pi}{\text{N}}(\text{n} - \text{kN})\right) \quad \text{n} \in \mathbf{Z}$$

In any other case, the role of the prefilter is to remove all the components outside $BL\left(-\frac{\omega_0}{2}, \frac{\omega_0}{2}\right)$. Indeed, if no prefilter is considered, then the Discrete-Time Fourier Transform of y

$$Y(e^{i\omega}) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(e^{i(\omega+2k\frac{\pi}{N})}) \quad \omega \in [-\pi, \pi)$$

contains the so called *spectral replica*. These frequencies overlap from the *base spectrum*, composed by all frequency with fixed $k = 0$, introducing distortion. This phenomenon is known as *aliasing* and occurs when the bandwidth $\omega_0 \geq \frac{2\pi}{N}$ but does not if $\frac{\pi}{N} < \omega_0 < \pi$. Hence, a sufficient condition for avoiding aliasing is that the sampling rate $\frac{1}{N}$ is greater than $\frac{\omega_0}{2\pi}$ [10]. Indeed, this value corresponds to twice the *Nyquist frequency*, the minimal sampling frequency that allows to restore the unmeasured values of the original signal, given its discrete sampled representation, according to the Nyquist-Shannon theorem [15].

3.2 Signals alignment

What described in the previous section has been applied to synchronize the signals under analysis.

As first step, it is required each SPN to have a constant sampling rate. Since this was not the case, each SPN has been interpolated in order to put it into a suitable form for further analysis. Both *linear* and *nearest* interpolation are used, leading to similar results. Hence *nearest* interpolation is chosen: the effect of this operation is first to discretized the temporal axis of each previously identified working cycles picking points with step length equal to the given SPN average time delta (described in Table 2.2) and then to slightly move along the temporal axes observations that were not at desired timestamps. Since signals refers to physical quantity measured with limited accuracy instruments and since the difference between original and new timestamps is in the order of milliseconds, this technique can be considered a good approximation for the behaviour of the real signals.

Once SPNs have been sampled with constant sampling rate, they are ready to be synchronized. There exists two possible choices that can be made: without losing information, it is possible to resample each signal to the higher sampling rate, each 20 milliseconds in this case. However, resampling data at the lowest rate (1 Hertz in the available data) is sometimes more appropriate since the extra points added upsampling could introduce noise and lead to distorted results [7]. In addition, also computational costs would be higher. The choice of downsampling the signal to the lowest rate is supported also by the analysis of the nature of the

available data: indeed, since they refer to physical quantity of a heavy-duty vehicle, they are not expected to have high-frequency variations but to change gradually, in a quite slow way.

Hence, according to the average sampling rate of each SPN, signals can be divided into three groups:

1. SPN 110, 182 and 975 are already sampled at 1 Hertz. Hence, no extra operation is needed.

2. SPN 94, 183, 190, 30000, 30694, 31391, 31800 are characterized by an integer downsampling factor to obtain the desired rate. Hence, it is sufficient to apply downsampling operator.

3. For SPN 524, 30789 and 32061 there exists no integer downsampling factor to achieve the desired rate. In this case, signals are first upsampled and then downsampled.

Upsampling is implemented by means of *zero padding* technique: it involves the addition of zeros to the original Discrete-Time Fourier Transform of the input sequence in order to increase the total number of input data samples in time domain, thanks to *Fourier* or *zero padding theorem* [16].

Zero padding can be defined as it follows [17]:

Definition 3.2.1 Zero padding *consists in mapping a signal x of length N to a signal of length M, with M not integer multiple of N, by adding a sequence of M zeros between the components $\frac{N}{2} - 1$ and $\frac{N}{2}$ if N even, $\frac{N-1}{2} - 1$ and $\frac{N-1}{2}$ if N is odd.*

Theorem 3 *An ideal bandlimited interpolation in time domain is equivalent to zero padding the original signal in the frequency domain [17]:*

$$\text{interp}_{\text{M}} = \text{IDTFT}(\text{Zero\_padding}_{\text{M}}(\text{X}))$$

Previous theorem can be interpret in terms of the *stretch* of a factor L operator [17], that maps a length N signal into a length $M = LN$ one as it follows:

$$\text{STRETCH}_{\text{L}} = \begin{cases} x(m/L), & L \mid m \\ 0, & L \nmid m \end{cases}$$

Hence it inserts $L - 1$ zeros between each element of the original signal.

Introducing the *ideal lowpass filtering operation in frequency domain* [17] with cut-off frequency $\omega = 2\pi\frac{M-1}{2N}$

$$\mathrm{F\_LP_{M,k}(X)} = \begin{cases} X(k), & -\dfrac{M-1}{2} \leq k \leq \dfrac{M-1}{2} \\ 0, & \dfrac{M+1}{2} \leq |k| \leq \dfrac{N}{2} \end{cases}$$

that sets the to zero the spectrum components $X(k)$ such that $k \notin [-\frac{M-1}{2}, \frac{M-1}{2}]$, it is possible to prove that the following theorem holds [17]:

Theorem 4 *Given a signal* $x \in \mathbf{C}^N$*,*

$$\mathrm{interp_M} = \mathrm{IDTFT(F\_LP_N(DFT(STRETCH_M(x))})$$

Hence, for obtaining a bandlimited interpolation of x by a factor M, the following sequence of operations can be performed: first the signal is stretched by a factor M, then its DTFT is computed and the ideal lowpass filter is applied to the result and finally the IDTFT is taken in order to go back to time domain.

For what concerns downsampling instead, the downsampling operation is preceded by a filter. The default anti-aliasing infinite impulse response filter provided by the decimate function in Scipy [1] is used. It is a order 8 Chebyshev type I filter, a low pass filter which exploits Chebyshev polynomials [18]. This choice is less complex with respect to use a finite impulse response filter and furthermore it introduces a lower delay.

Applying the described procedure separately for each SPN and working cycle, the resulting dataset is shown in figure 3.7, where the row index indicates the timestamp of each observation in unix format while each column contains the measurement for the considered SPN.

[1] www.scipy.org

timestamp	524	30789	32061	110	182	975	183	190	30000	30694	31391	31800	94
1573144720000	1.0	189.060356	0.000000	22.0	11320.5	100.0	4.544996	638.309462	36.525690	86.590287	0.0	16260.709952	144.0
1573144721000	1.0	189.355200	0.000000	22.0	11320.5	100.0	4.553227	635.491827	36.688077	86.788810	0.0	16260.709894	144.0
1573144722000	1.0	188.806078	0.000000	22.0	11320.5	100.0	4.513245	636.413379	36.205762	86.555376	0.0	16260.709979	144.0
1573144723000	1.0	189.064634	0.000000	22.0	11320.5	100.0	4.458951	638.268883	35.582337	86.619425	0.0	16260.710013	148.0
1573144724000	1.0	188.779694	0.000000	22.0	11320.5	100.0	4.495576	639.881501	35.535615	86.718170	0.0	16260.710016	148.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1586968191000	1.0	213.117276	0.792327	74.0	11751.5	100.0	3.862486	830.204598	29.786025	96.581031	0.0	16260.709960	148.0
1586968192000	1.0	212.986015	0.792673	74.0	11751.5	100.0	4.000215	837.175063	28.439597	96.538470	0.0	16260.709999	148.0
1586968193000	1.0	212.831316	0.791908	74.0	11751.5	100.0	3.856756	838.893066	27.390466	96.500766	0.0	16260.710016	148.0
1586968194000	1.0	212.709839	0.791029	74.0	11751.5	100.0	3.686585	838.932472	27.124967	96.501883	0.0	16260.710003	148.0
1586968195000	1.0	212.654637	0.790868	74.0	11751.5	100.0	3.686585	838.932472	27.124967	96.501883	0.0	16260.710003	148.0

294265 rows × 13 columns

Figure 3.7: First rows of resampled dataset

Chapter 4

Time Series Analysis

The term *time series* denotes a sequence of quantitative observations about a system or process made at successive points in time [19]. If the time series is *regular*, then points are equally spaced in time. As previously introduced, the frequency of observations depends in general on applications and on the described variable.

From a more probabilistic point of view, a time series can be defined from the notion of stochastic process [20]:

Definition 4.0.1 *A stochastic process is a parameterized collection of random variables* $\{X_t\}_{t \in T}$ *defined on a probability space* $(\Omega, \mathcal{F}, \mathbf{P})$ *and assuming values in* $\mathbf{R}^n$.

From the previous definition, it follows that a time series can be seen as a set of observations $x_t,\ t \in T$ that represent the realizations of the random variable X_t, each recorded at a specific instant of time t.
The time series is said to be *discrete-time* if the set T is a discrete set, *continuous-time* otherwise.

If the random variables take values in $\mathbf{R}$, then the time series is said to be *univariate*. It is said to be a *multivariate* time series otherwise.

Since a time series is made up of one or more measurable characteristics of an individual entity taken at multiple points in time, resampled SPNs have the typical structure of a time series.
As first step, they can be then analysed separately, as a univariate time series, taking as *individual entity* the single parameter monitored by a given SPN. But each SPN can also be treated as a component of a tuple containing, at a given timestamp, the set of observation recordered for all the monitored parameters, considering the vehicle itself as an *individual entity*.

Indeed, a l-dimensional multivariate time series of length n is a set of points

$$X = \{X(t_k) \in \mathbf{R}^l, k = 1, \ldots, n\}$$

such that

- for $j < k \;\Rightarrow\; t_j < t_k$
- $t_k \in T = [a, b], \quad \forall k = 1, \ldots, n$
- $X(t) \in \mathbf{R}^l, \; \forall t \in T$

If each SPN corresponds to a dimension of $\mathbf{R}^l$, methods for multivariate time series analysis can be applied to the dataset shown in Figure 3.7.

4.1 Univariate time series analysis

Definition 4.1.1 *Let* $\{X_t\}$ *be a univariate time series such that* $\mathbf{E}(X_t^2) < \infty$. *Then its* mean function *is defined as*

$$\mu_x(t) = \mathbf{E}(X_t) \quad \forall t \in \mathbf{R}$$

However, since usually in applications only a limited number of observations $x_1, x_2, \ldots, x_n$ is available, the *sample mean* is introduced,

$$\overline{x} = \frac{1}{n} \sum_{t=1}^{n} x_t.$$

Definition 4.1.2 *The* covariance function *is*

$$\gamma_X(t, s) = \mathrm{Cov}(\mathrm{X_t}, \mathrm{X_s}) = \mathbf{E}\left(\mathrm{X_t} - \mu_{\mathrm{X}}(\mathrm{t})\right)\left(\mathrm{X_s} - \mu_{\mathrm{X}}(\mathrm{s})\right)$$

$\forall r, s \in \mathbf{R}$

Finally, its *correlation* is given by

$$\rho(t, s) = \frac{\mathrm{Cov}(\mathrm{X_t}, \mathrm{X_s})}{\sigma_t \sigma_s}$$

Important properties of time series are *ergodicity* and *stationarity* [21], [20].
A time series is said to be *erogidc* if the sample moments computed on a limited number of observations converge, as the number of observations grows, to the corresponding moments of the population [21]. This property is obviously satisfied if, for example, the expectations and the variances are constant $\forall t$.
Instead, a time series is said to be *stationary* if its internal structure does not

change over time [20]. More formally, a time series is said to be *strictly* stationary if the joint distribution of $(x_{t_1}, x_{t_2}, \ldots, x_{t_n})$ is equal, for every $h > 0$, to the joint distribution of $(x_{t_1+h}, x_{t_2+h}, \ldots, x_{t_n+h})$ [20].

However this is a strong condition, so in practise the following less restricting concept of stationarity is required [21]:

Definition 4.1.3 $\{X_t\}$ *is said to be* weak *stationary or* stationary in the second moments *if it is mean and variance stationary.*
A process is said to be mean *stationary if*

$$\mathbf{E}(X_t) = \mu_t = \mu$$

that is, if it is characterized by constant mean.
A process is said to be variance stationary *if*

$$\mathrm{Var(X_t)} = \mathbf{E}((\mathrm{X_t} - \mu_\mathrm{t})^2) = \sigma^2 = \gamma(0)$$

is constant and finite for every t.

If the moments are well defined, then stationarity implies weak stationarity [20].

If $\{X_t\}$ is (weak) stationary, the covariance between X_{t+h} and X_t can be written as

$$\mathbf{Cov}(X_{t+h}, X_t) = \gamma_X(t+h, t) = \gamma_X(h,0) = \gamma_x(h) \ \forall t \in \mathbf{R}$$

$\gamma_X(h)$ is called *autocovariance function* (ACVF) at lag h [20].

In order to make comparable the strength of dependence over time, the autocovariance function can be normalized, obtaining the *autocorrelation function* (ACF) [21]:

Definition 4.1.4 *The* autocorrelation function *at lag* h *is defined as*

$$\rho(h) = \frac{\mathbf{E}\left[(X_t - \mu)(X_{t+h} - \mu)\right]}{\mathbf{E}\left[(X_t - \mu)^2\right]} \quad \forall t \in \mathbf{R}$$

It is called autocorrelation because it represents the correlation between random variables belonging to the same underlying stochastic process that, in the case of (weakly) stationary process, depends only on the time lag h but not on time index. It follows from the given definition that $\rho(0) = 1$ and, thanks to variance symmetry, $\rho(-h) = \rho(h)$ [21].
Limit bounds for the autocorrelation function are given by

$$-1 \leq \rho(h) \leq 1 \ \forall h.$$

In applications, thanks to ergodic assumption, variance and autocovariances can be estimated as [21]

$$\hat{\gamma}(0) = \frac{1}{n}\sum_{t=1}^{n}(x_t - \hat{\mu})^2$$

$$\hat{\gamma}(h) = \frac{1}{n}\sum_{t=1}^{n-h}(x_t - \hat{\mu})(x_{t+h} - \hat{\mu}) \quad h = 1, \ldots, n-1$$

Previous expressions define consistent estimators for variance and autocovariances. A consistent estimator for autocorrelation function is given by the *sample autocorrelation function* [21]

$$\hat{\rho}(h) = \frac{\sum_{t=1}^{n-h}(x_t - \hat{\mu})(x_{t+h} - \hat{\mu})}{\sum_{t=1}^{n}(x_t - \hat{\mu})^2} = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \quad h = 1, \ldots, n-1$$

Commonly, the estimated autocorrelation function is computed for different values of h and plotted together with confidence interval to evaluate statistical significance of correlations. The serial correlation coefficients for a certain number of consecutive lags is called *correlogram* [19].
Because of symmetry, it is sufficiently then to consider $h \geq 0$.
Present and future observations are in the same direction if a time series shows positive autocorrelation, opposite direction for negative autocorrelation and it is difficult to find temporal dependencies if it is close to zero.
Furthermore, correlograms can be used to identify specific components of a time series [19]: for example, if data contains a trend on the long run, then the correlogram will exhibit a slow decay as h increases. Analogously, it is possible to verify if data has periodic components searching for a similar behaviour in autocorrelation function plot. There exist several tests to determine if the observed autocorrelation is statistically significant, such as the Ljung-Box test [19]. The null hypothesis H_0 is that the time series consist of random variations, while the alternative one H_a is that observations are not independently distributed but there is a serial correlation. The test statistic is

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k}$$

where n denotes the sample size and h is the number of lags to be tested.
Under H_0, Q is asymptotically following a Chi-square distribution with h degrees of freedom. Hence, for a given and a priori fixed significance level α, there is enough evidence to reject H_0 if

$$Q > \chi^2_{1-\alpha,h}$$

where $\chi^2_{1-\alpha,h}$ denotes the $1-\alpha$ quantile of the chi-squared distribution with h degrees of freedom.

Even if correlogram inspection can be useful to identify dependencies, it should be considered that consecutive lags could be statistically dependent: indeed, the correlation between two observations with a temporal lag $h > 1$ could be influenced by the correlation of these observations with the intermediate ones. This means that time series tends to carry information from previous observations. For this reason, the notion of autocorrelation function is extended removing the intermediate dependencies, obtaining the *partial autocorrelation function* [19]:

Definition 4.1.5 *The* partial autocorrelation function *at lag h is defined as*

$$PACF(h) = \frac{\mathbf{E}\left[(X_t - \mu)(X_{t+h} - \mu) \mid x_{t+1}, \ldots, x_{t+h-1}\right]}{\mathbf{E}\left[(X_t - \mu)^2 \mid x_{t+1}, \ldots, x_{t+h-1}\right]} \quad \forall t \in \mathbf{R}$$

4.1.1 Trend and seasonality analysis

Autocorrelation and partial autocorrelation function can be useful tools for finding the systematic patterns contained in a time series and then for inspecting stationarity. Indeed, systematic patterns in a time series can be described by means of the three main internal characteristic: trend, seasonality and cyclical components. Usually the aim is to estimate these components, using autoregressive models, to split the original time series into different components describing the general underlying behaviour of data [22]. However, the identification of trend, seasonality and cyclical components can be difficult because they can be obfuscated by random noise (also called *unexpected variations*), a stationary time series representing the irreducible error. It reflects unexpected stochastic variations that can not be predicted by a mathematical model. For this reasons, they are also called *residuals*.
General trend can be defined as an upward or downward movement in the long run. Sometimes, it can be difficult to detect because it can be obfuscated by other components.
Seasonality represents instead repetitive and periodic variations.
Finally, cyclical changes are similar to seasonality but the occurrences of repetitive movements is less frequent and they could be not characterized by a fixed period. This component is usually considered as a part of trend.

A time series analysis can be seen as the series decomposition into trend F_t, seasonal S_t and irregular component ε_t [19].

More in details, if the model is *additive*, then it can be written as

$$x_t = F_t + S_t + \varepsilon_t$$

If instead it is multiplicative,

$$x_t = F_t S_t \varepsilon_t$$

An additive model is appropriate when seasonal variations are independent on the series value. If instead the amplitude of variations is proportional with the series value, a multiplicative model is more appropriate. However, by applying a logarithmic function it is possible to rewrite a multiplicative model as an additive one.
For what concerns trend analysis, the aim is to estimate a component that is, in the long run, monotonous. It could be not that easy to identify the trend component, if time series contains some observations that can be considered outliers or errors. For this reason, what is typically done for estimating trend component is smoothing the time series [19]. A non-parametric method for trend estimation is the *moving average*, defined as [19]

$$\hat{F}_t = \frac{x_{t-k} + x_{t-k+1} + \ldots + x_t + x_{t+1} + \ldots + x_{t+k}}{2k+1}$$

that is the average value considering only observations in the range $t \pm k$. k is chosen basing on data periodicity, known or inferred during exploratory analysis. Replacing each observation with the mean of the $2k$ nearby observations, random noise is removed. From the definition of the moving average it is possible to see that it is characterized by its order $m = 2k + 1$, that shows an asymmetry. To obtain a symmetric structure, it is possible to consider the second order moving average, defined as [19]

$$2 \times \hat{F}_t^2 = \frac{\hat{F}_t^2 + \hat{F}_{t+1}^2}{2}$$

where

$$\hat{F}_t^2 = \frac{x_{t-1} + x_t}{2}$$

There exists also a weighted version of moving average, assigning to observation close to t higher weights whereas lesser to farther away ones, that helps to obtain a better smoothing.

Seasonal dependency is instead defined as correlation dependency at lag k measured by the autocorrelation function. For this reason, it can be identified from the correlogram and from the partial autocorrelation plot [23].

4.1.2 SPN autocorrelation analysis

The first step to analyse time series consists in visualizing the available data points. According to the type of behaviour of SPNs all over the available observations, it is possible to made a distinction in different groups. The described SPNs for each group are shown in Figure 4.1.

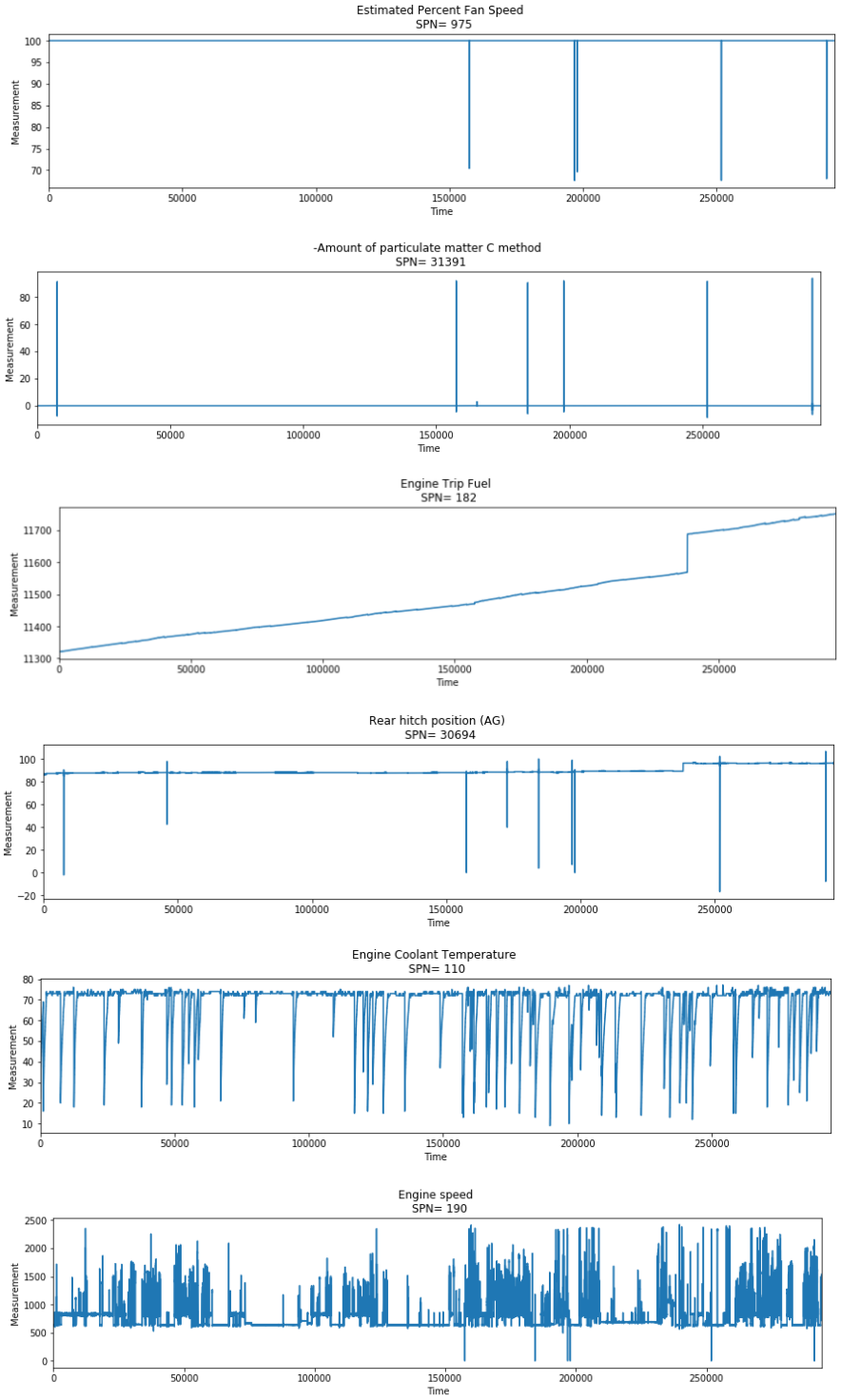

Figure 4.1: Time series plot for representative SPNs

The first one is represented by SPN 975, corresponding to the estimated percent fan speed. It does not exhibit any sort of trend or seasonality since it is fixed on the same value except in really short periods of time. Some of SPNs in this group could seem categorical variables, but from a closer look on their values, variations typically associated to real valued variables can be seen. It is not clear if they are actually due to the nature of the parameter or if caused by errors in measurements or transmission. Anyway, probably they are not enough informative for further analysis, but they can be used to check, once the results have been obtained, if a change in their value corresponds to some specific events. Its correlogram is expected to show high correlation at lags corresponding to the variations and not significant correlations in the remaining lags.

The second group is represented by SPN 182, corresponding to the engine trip fuel. Because of the nature of the SPN, it looks like to have a trivial very smooth and linear upward trend, with the exception of a single evident point. This point of discontinuity corresponds to the first observation taken in March 18, 2020, while the last observation before it was dated back to January 22, 2020, meaning that in the meanwhile the vehicle was used but no data was collected, intentionally or for some failures.
Since it is an incremental value, its correlation is expected to be significant for multiple consecutive lags. An analogous upward trend is shown by SPN 30694, corresponding to the rear hitch position. Its identification is more difficult with respect to the previous case because its trend is obfuscated by other components. However, by smoothing the signal using a moving average, it is possible to see its long term increasing behaviour. Its shape suggests that in a first period the vehicle worked with the rear hitch in lower positions with respect to more recent observations.

To obtain stationarity for both signals it is sufficient to apply a first order differencing technique and replace the n points of each series with $n - 1$ points corresponding to the differences between consecutive values. But since they are considered, in the following, as components of the same multivariate time series, the same procedure should be applied to each SPN. However SPNs 182 and 30694 are probably not significant for profiling usage patterns, hence it is preferable to not consider them in the following analysis and keep other signals in their original form, preventing overfitting.

Finally, the last group can be represented by the engine speed (SPN 190) or by the previously introduced engine coolant temperature (SPN 110). From their plot it is immediate to recognize seasonality inside data: indeed, there are a lot of repeated movements reflecting the repetitive behaviour of working cycles. In this case, the correlogram is expected to have some peaks and not significant correlation for many consecutive lags. The expected difference between the two correlograms is the frequency of peaks: indeed, the engine coolant temperature has cyclical

behaviour repeated once per working cycle, while the engine speed is more likely to cyclical vary also inside a given working cycle.

The correlograms obtained from data are shown in Figure 4.2. The behaviour is quite coherent with the expected one. Indeed, the first two SPNs show peaks at lags corresponding to signal variations from their fixed value. SPN 182 and 30694 correlograms confirm the trivial dependence among consecutive values. The last two SPNs have an initial high autocorrelations and some peaks that reflect their cyclical behaviour. However, most of them are not statistically significant according to the 95 percent confidence interval.

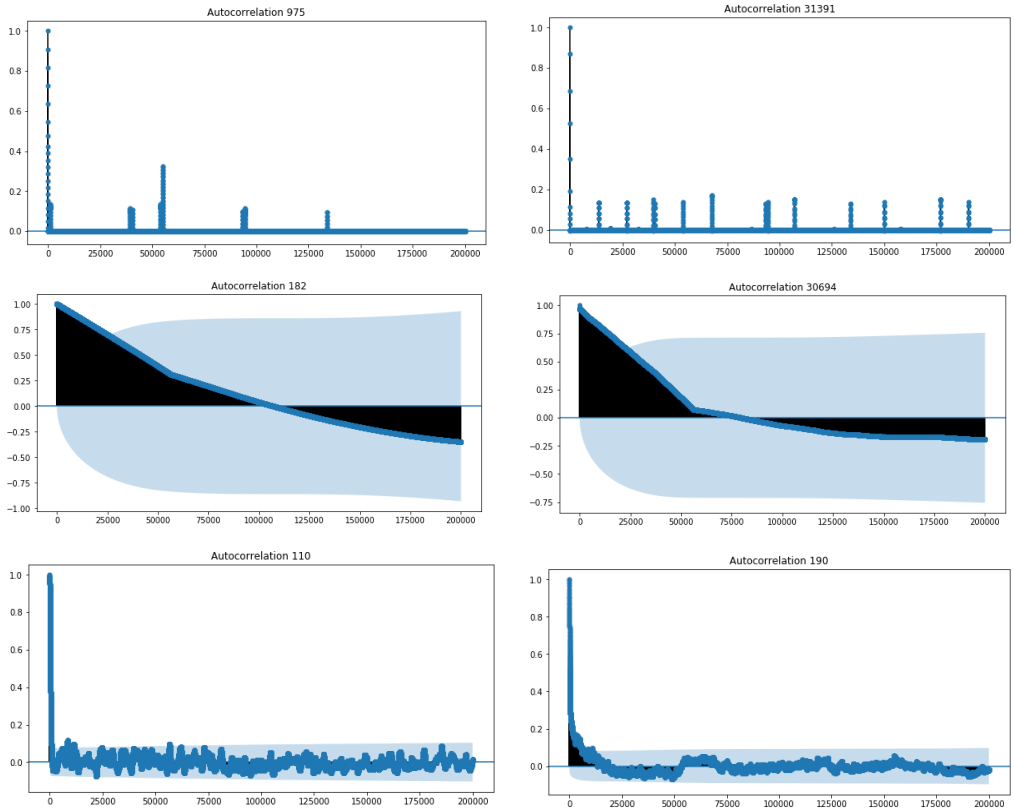

Figure 4.2: Correlograms for representative SPNs

Finally, the partial autocorrelation function is shown in Figure 4.3. As it is possible to see, most of the higher autocorrelations shown in Figure 4.2 are due to influence by intermediate values. Indeed, partial autocorrelation is quite low for all SPNs, with the exception of the first few lags.

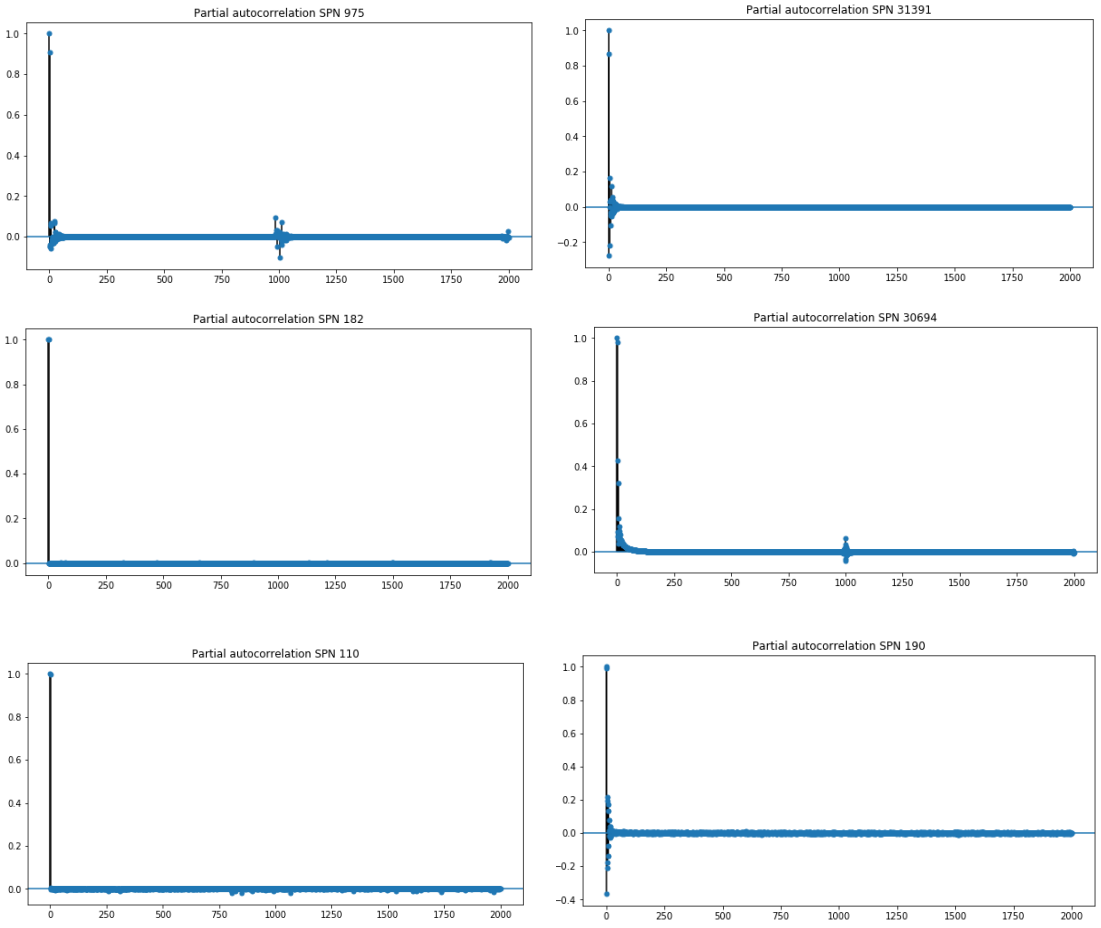

Figure 4.3: Partial Autocorrelation plots for representative SPNs

To summarize:

- There is a group of SPNs (namely SPN 182 and 30694) with trivial trends that do not add much information for the analysis. Hence, they can be ignored since they are not significant for usage patterns profiling.

- There is a group of SPNs that are fixed on a single value with the exception of really short intervals of time: the time deltas between the timestamps in which the signal changes its value correspond to lags characterized, in the

correlogram, by a high and significant correlation, as can be seen for SPN 975 and 31391 in Figure 4.2. They are probably not enough informative for the following analysis, but they can be used in some specific applications since a change in their values could represent a specific but for the moment unknown event or a malfunction tell-tale.

- The largest group of SPNs is characterized by a strong seasonality due to working cycles. As it is possible to be seen for SPN 110 and 190 in Figure 4.2, there are some oscillations in the correlogram reflecting seasonality, but they are not significant since they are almost at every lag contained inside the 95% confidence interval. In both the presented cases, it is possible to recognize seasonality in the data inspecting Figure 4.1. For the engine coolant temperature the interpretation is quite simple: when the vehicle is turned on and the first observation is taken, the engine is cold and then, going on with the work, it warms up. For this SPN, the cyclical component is repeated once for working cycle. For what concerns the engine speed instead, it is possible to notice that there are several repeated movements but in shorter periods with respect to the previous case: indeed, when the vehicle is turned on and it starts to move, the engine speed is expected to rapidly grow and then decreasing when the cruising speed is reached or when the gear is changed. Hence, the cyclical component is repeated several times in the same working cycle. For analogous reasoning, the same seasonality behaviour is shown by the engine speed (SPN 183) and the engine percent load (SPN 92).
- Most of SPNs parameters have low autocorrelations with wide lags, meaning that the time series does not exhibit a notable autocorrelation persisting for long lags. Excluding SPNs 182 and 30694 for the reasoning explained above, analysing up to 200000 consecutive lags, it is possible to see that all the remaining SPNs show not significant correlation after 10000 lags.

4.2 Multivariate time series analysis

Once SPNs have been individually studied, it is possible to see if there exist dependencies among different parameters and how they influence each other since they are part of the same system.
This can be done first by analysing cross-correlation of SPNs pairs and then building a vector autoregressive (MVAR) model for multivariate time series to investigate Granger causality.

4.2.1 Cross-correlation analysis

The concept of cross-correlation has been developed both in signal processing and statistics.
In the former, it can be used to transform one or more signals in order to inspect them in an altered perspective [24]. For instance, the cross-correlation plots can make easier the identification of hidden signals in data.
From a statistical point of view instead, cross-correlation can be seen as a measure of association between time series. More in details, it is a common practice to shift one curve with respect to the other in order to inspect dependencies even if the change of a parameter affects the other with a certain delay. The number of data points that the first signal is shifted is called lag [24]. Hence, the cross-correlation is defined as it follows [25]:

Definition 4.2.1 *Given two time series x_t and y_t both composed by N observations with mean respectively μ_x and μ_y, the* cross-correlation *at lag h is defined as*

$$\tau_{x,y}(h) = \frac{\sigma_{x,y}(h)}{\sqrt{\sigma_{x,x}(0)}\sqrt{\sigma_{y,y}(0)}}$$

where $\sigma_{x,y}(h)$ is the cross-covariance *function,*

$$\sigma_{x,y}(h) = \frac{1}{N-1}\sum_{i=1}^{N}(x_{t-h} - \mu_x)(y_t - \mu_y)$$

From previous definition it is possible to note that, since $\sigma_{x,x}(0) = \sigma_x^2$ and $\sigma_{y,y}(0) = \sigma_x^2$ are the variances of each time series,

$$\tau_{x,y}(0) = \frac{\sigma_{x,y}(0)}{\sigma_x \sigma_y}$$

is the *Pearson correlation* between the two variables [25].

The Pearson correlation between each pair of SPNs is graphically displayed in Figure 4.4, exploiting the symmetry of the correlation function.

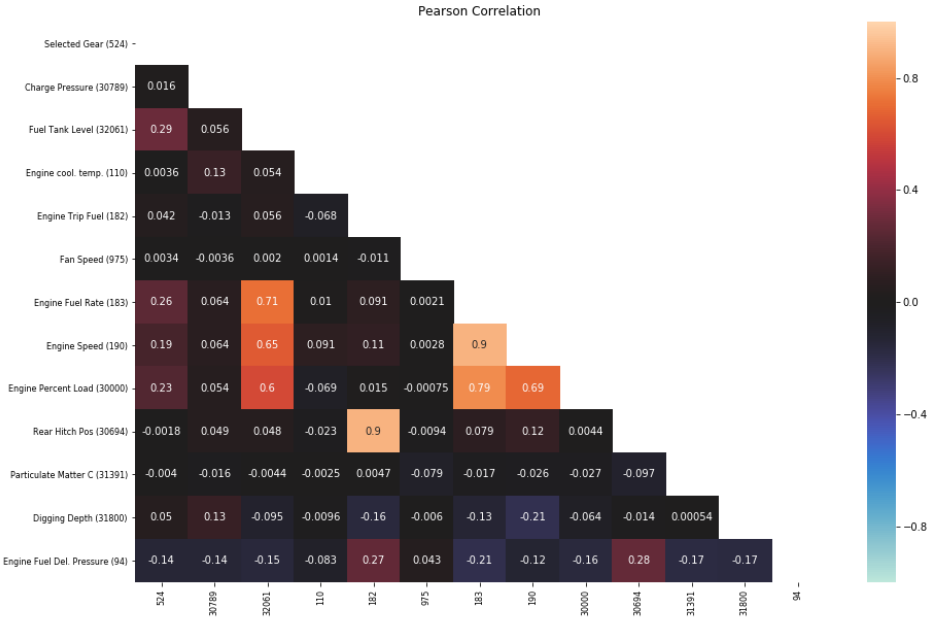

Figure 4.4: Pearson correlation between SPNs

As it is possible to see, there is a group of strongly positive correlated features composed by engine parameters, namely SPN 183, 190 and 30000, monitoring respectively the engine fuel rate, the engine speed and the engine percent load. They are also strongly correlated to another parameter, not directly connected to the engine, the tank fuel level (SPN 32061). Since they provide almost the same amount of information, to avoid multicollinearity issues SPNs 183, 30000 and 32061 can be removed, considering only the engine speed in the following analysis.

The same reasoning holds for spn 30694 and 182, representing respectively the rear hitch position and the engine trip fuel. However, since this two features exhibit a trivial trend as it was previously noticed, both of them should not be considered.

The remaining SPNs are characterized instead by low correlations.

However, computing correlation separately for each SPN is not sufficient since data are heterogeneous [26], meaning that is possible to identify different groups of variables. Indeed, there are engine parameters (namely SPN 94, 110, 182, 183, 190, 30000), parameters referring to vehicle attachments and working accessories (namely 30694, 30789, 31800), while the transmission selected gear, the estimated percent fan speed, the amount of particulate C matter method and the fuel tank level can be treated separately. Analysing the correlation between groups of parameters,

called *between-group correlation*, it is possible to study how a group of variables affects another one. The between-group correlation can be obtained computing the average of the variables belonging to the same group and then computing the Pearson correlation between the obtained average series of different groups [26]. Results are shown in Figure 4.5.

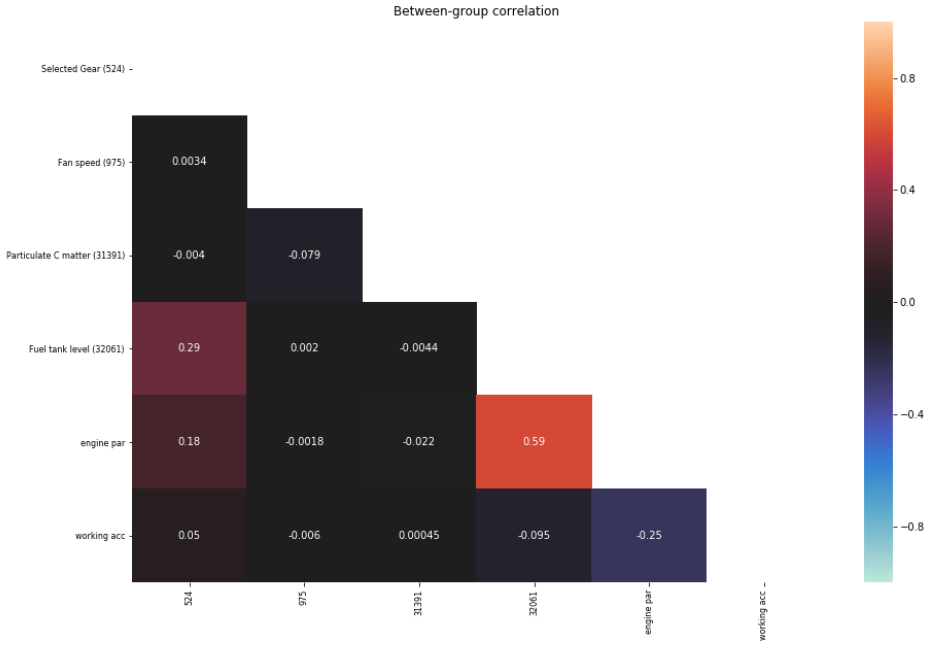

Figure 4.5: Between-group correlation
The figure represent the correlation between engine parameters (denoted by *engine par*), the group referring to vehicle's attachments and working accessories (denoted by *working acc*) and the transmission selected gear, the estimated percent fan speed, the amount of particulate C matter method and the fuel tank level.

As it is possible to see, the engine parameters are strongly positively correlated with the fuel tank level as already noticed as result of Person correlation in Figure 4.4. However, the between-group correlation analysis highlights a previous not noticed negative correlation between engine parameters and the SPNs referring to vehicle's attachments and working accessories: this could seem a contradiction, since to higher charge pressures and digging depths should corresponds higher engine percent load and engine speed values. However, according to domain experts, the result is coherent with the tasks performed by the vehicle under analysis: indeed, as previously introduced, the vehicle is not actually working while data are collected. Hence such a behaviour could be due to the fact that, when the vehicle is moving or

moving faster, the rear hitch is raised to higher positions. However, for a actually working vehicle, these groups are expected to be probably positively correlated.

For strictly positive lags instead, the behaviour of cross correlation is shown in figure 4.6. Due to computational costs, it is computed and displayed for 2000 consecutive lags, corresponding to a maximum delay of about half an hour. This value is adequate to describe cross correlation between parameters since it is slightly higher than the average length of working cycles. Furthermore, the cross correlation for a limited portion of the dataset has been analysed also for lags greater than 2000, showing correlations close to zero also for higher lags.

The first one represents the cross correlation between SPN 190 and 183, but it also summarizes the behaviour of the cross-correlation for all the other highly correlated parameters in the same group. As it is possible to see, correlation at lags 0 is quite strong and persists for several lags.

The second one describes the correlation between the parameters which constitute the second group of highly correlated features. In this case, the elevate correlation persists for all shown lags.

The last two plots summarize the general behaviour of not correlated SPNs: indeed, in this case the cross correlation tends to 0 as the number of lags increases or it is about constant on its initial negligible value.

To summarize cross correlation general behaviour, it is possible to see that for most pairs with positive (resp. negative) correlation, the cross correlation exhibits a downward (resp. upward) trend as the number of lags increases and tends to 0, with different slopes depending on the considered pair. Sometimes it shows smooth and not significant oscillations. In few isolated cases the cross correlation is instead almost constant. In any case, it is possible to say that if the correlation at lags 0 is negligible, then it is also negligible for higher lags.

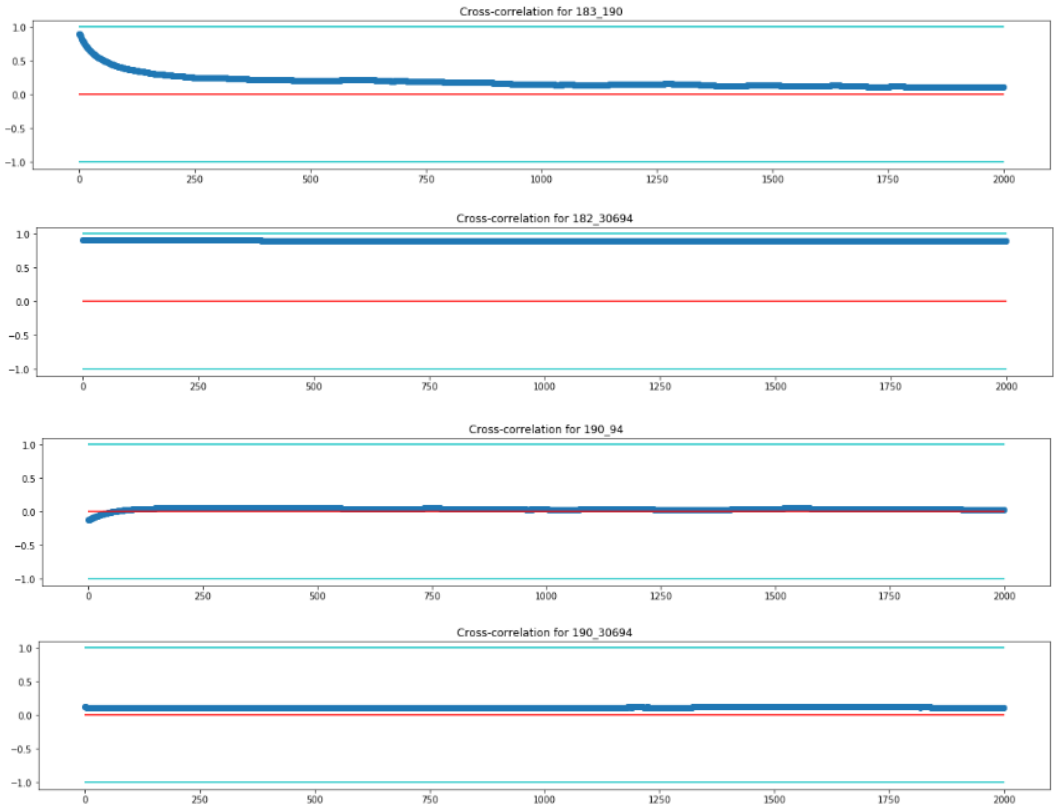

Figure 4.6: Cross correlation plots for representative SPNs
The red line represents the 0 value, while the light blue ones denote the lower and the upper limit of cross-correlation, respectively -1 and $+1$.

4.2.2 Granger causality analysis

Finding causality between components of a multivariate processes is of particular interest. The testable definition of causality was introduced by Granger and first applied in economics studies [15]. The definition is based on the predictability of a term of time series thanks to the information contained in past terms of another one. In order to give the formal definition, it is necessary to introduce the concept of multivariate autoregressive model (MVAR):

Definition 4.2.2 *Given a l-dimensional multivariate time series of length n $X = X(t_k) \in \mathbf{R}^l$, $k = 1, \ldots, n$, where $X(t_k) = (x_{1,t}, x_{2,t}, \ldots, x_{k,t})$, a* multivariate autoregressive model *of order p is a linear relationship that allows to write $X(t_k)$ as a linear combination of the p previous values of the multivariate series:*

$$X(t_k) = A_1 X(t_{k-1}) + A_2 X(t_{k-2}) + \ldots + A_p X(t_{k-p}) + \epsilon(t_k). \tag{4.2}$$

$\epsilon(t_k) = (\epsilon_1(t_k), \epsilon_2(t_k), \ldots, \epsilon_l(t_k))$ is a vector of noise process samples at time t_k representing the error term. Its covariance matrix V can be expressed as

$$V = \epsilon(t_k)\epsilon(t_k)^T = \begin{pmatrix} \sigma_1^2 & 0 & \ldots & 0 \\ 0 & \sigma_2^2 & \ldots & 0 \\ \vdots & & & \vdots \\ 0 & \ldots & 0 & \sigma_l^2 \end{pmatrix}$$

where σ_i^2 is the variance of the ith component of $\epsilon(t_k)$, $\forall k$. V represents the residual variance not explained by the model. In applications, it is possible to obtain only an approximation of V, denoted by $\hat{V}$.

The MVAR model coefficient for each lag h is a $l \times l$ matrix that can be written as

$$A_h = \begin{pmatrix} A_{1,1}(h) & A_{1,2}(h) & \ldots & A_{1,l}(h) \\ A_{2,1}(h) & A_{2,2}(h) & \ldots & A_{2,l}(h) \\ \vdots & & & \vdots \\ A_{l,1}(l) & \ldots & A_{l,l-1} & A_{l,l}(h) \end{pmatrix}$$

For all $h = 1, \cdots, p$, this matrix components need to be estimated. However, before starting the fitting procedure, it is necessary to slightly modify the original time series components: indeed, each signal should be standardized, subtracting from each observation the signal temporal mean and dividing by its temporal standard deviation. Then, the estimation of model parameters is an extension of the classical technique for univariate autoregressive model parameters estimation based on the Yule-Walker algorithm [15], requiring the computation of the correlation matrix R of the system up to lag p. In order to do that, both side of equation (4.2) are multiplied by $X(t_{k+s})$ for $s = 0, \ldots, p$ and expectations are taken:

$$R_{i,j}(s) = \frac{1}{N} \sum_{k=1}^{N} x_{i,t} x_{j,t+s}$$

Assuming that the noise is indipendent on the series, the following set of linear equations, called *Yule-Walker* equations, is obtained:

$$\begin{pmatrix} R(0) & R(-1) & \ldots & R(p-1) \\ R(1) & R(0) & \ldots & R(p-2) \\ \vdots & \vdots & & \vdots \\ R(1-p) & R(2-p) & \ldots & R(0) \end{pmatrix} \begin{pmatrix} A(1) \\ A(2) \\ \vdots \\ A(p) \end{pmatrix} = \begin{pmatrix} R(-1) \\ R(-2) \\ \vdots \\ R(-p) \end{pmatrix}$$

and

$$\hat{V} = \sum_{j=0}^{p} A(j)R(j)$$

Solving these equations, coefficients can be finally estimated.

To find the optimal model order instead, several criteria can be used. They are commonly based on the minimization of a cost function consisting of two terms: one representing the reward for minimizing the residual variance while the second one is a penalization term for too high model orders. For example, the Akaike information criterion (AIC) at order p is defined as

$$AIC(p) = \log[det(\hat{V})] + 2\frac{pl^2}{N}$$

The Bayesian information criterion (BIC) instead is based on the minimization of the function

$$BIC(p) = \log[(\hat{V})] + \log(N)\frac{pl^2}{N}$$

while the function for the Hannan-Quinn information criterion (HQIC) is

$$HQIC(p) = \log[(\hat{V})] + 2\log(\log(N))\frac{pl^2}{N}$$

and finally, for the Akaike's Final Prediction Error, the minimization function is given by

$$FPE(p) = \log[(\hat{V})] + \left(\frac{N - pl + 1}{N - pl - 1}\right)^l$$

This procedure is applied to build a MVAR model fitting the available data. The selected lags according to each criterion are summarized in Table 4.1.

Criterion	Optimal lag (p)
AIC	80
BIC	28
HQIC	80
FPE	38

Table 4.1: Optimal MVAR order according to AIC, BIC, HQIC and FPE

Lower order models are in general preferred in order to prevent overfitting. Hence, a order equal to 28 is selected to fit the model and to study Grenger causality.

Since Granger causality was originally defined for the bidimensional case, lets consider two univariate time series $x(t_k)$ and $y(t_k)$, $k = 1, \ldots, n$. However, the

same arguments apply also to higher dimensional situations and can be easily generalized.

To predict a value of $x(t)$ it is possible to build a model using p previous values of x only, getting an error ϵ:

$$x(t_k) = \sum_{j=1}^{p} A'_{1,1}(j)x(t_{k-j}) + \epsilon(t_k)$$

However, if the two signals are considered as components of a multivariate time series, from equation (4.2) it is possible to see that the value of $x(t)$ can be predicted using also the p previous observations of y, obtaining another prediction error ϵ_1:

$$x(t_k) = \sum_{j=1}^{p} A_{1,1}(j)x(t_{k-j}) + \sum_{j=1}^{p} A_{2,1}y(t_{k-j}) + \epsilon_1(t_k) \tag{4.3}$$

Then, y is said to *Granger cause* x if the variance ϵ_1 is smaller than the variance ϵ.

From this principle, formulated by Granger, it is possible to define the Granger causality index [15]

$$GCI_{y \longrightarrow x} = \log \frac{\epsilon_1}{\epsilon}$$

This definition can be extended to quantify directed influence from a component x_j to another one x_i of a l multivariate time series considering l and $l-1$ dimensional MVAR models [15]. Indeed, first the model is fitted on the whole set of components, obtaining a given residual variance $\hat{V}_{i,l}(t)$. Then, a $l-1$ dimensional model is fitted removing from the components the jth one, obtaining a variance $\hat{V}_{i,l-1}(t)$. The Granger causality index is then defined as

$$GCI_{j \longrightarrow i}(t) = \log \frac{\hat{V}_{i,l}(t)}{\hat{V}_{i,l-1}(t)}$$

From definition, it follows that $GCI \leq 0$ since the variance of the l-dimensional system is always lower than the residual variance of a $l-1$-dimensional one.

To test if a component x Granger-causes another one y, a Granger causality test can be performed. Considering the expression in equation (4.3), a Granger test is a F-test carried out to verify if there is enough evidence to reject the null hypothesis $H_0 : A_{2,1} = 0$, that is non-causality, while the alternative hypothesis is that y Granger causes x. The unrestricted model will be the one including also previous observations of y, while the restricted one will be the one including only

observations of x. Hence, it is possible to compute the F statistic as

$$F = \frac{RRSS - URSS}{URSS}\left(\frac{n-p}{p}\right)$$

where $RRSS$ is the restricted model's residual sum of squares, $URSS$ is the unrestricted model's residual sum of squares, n is the sample size and p is the number of considered lags. Under H_0, F follows a Fisher distribution $\mathcal{F}(p, n-p)$. Hence, for a given and a priori fixed significance level α, there is enough evidence to reject H_0 if

$$F > \mathcal{F}_{1-\alpha,\ p,\ n-p}$$

where $\mathcal{F}_{1-\alpha,\ p,\ n-p}$ denotes the $1-\alpha$ quantile of the Fisher distribution with $(p, n-p)$ degrees of freedom.

As previously introduced, the Granger causality test is performed considering a 28-order MVAR model on each pair of SPNs. The p-values of the test results, considering as significant level $\alpha = 0.05$, are displayed in Figure 4.7.

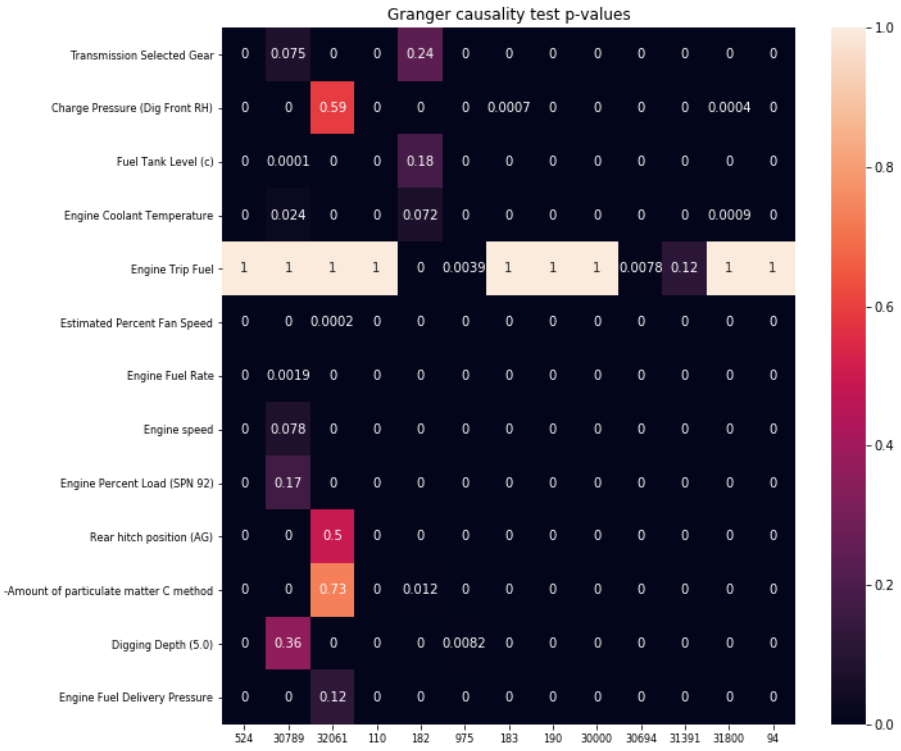

Figure 4.7: p-value of Granger causality test for MVAR(28)
The element (i, j) of the displayed matrix represent the p-value result at significant level $\alpha = 0.05$ of the Granger test with null hypothesis H_0: *"j does not Granger cause i"*.

In a statistical hypothesis test, the p-value represents the probability that a value at least as large as the test statistic will have occurred if H_0 is true [27].

The element (i, j) of matrix dispalyed in Figure 4.7 is the p-value of the test with null hypothesis H_0: *j does not Granger-causes i*. As it is possible to see, there is enough evidence to reject H_0 for most SPNs pairs, meaning that a statistically significant causality among SPNs at the considered lag can be assumed until there is proof to the contrary. However, there are also some cases in which the test fails to reject H_0. For example, the transmission selected gear is caused by all parameters related to the engine but there is no enough evidence in data for saying that it is Granger caused also by the charge pressure and the fuel tank level. Analogously, it is not possible to say that the engine trip fuel is caused by the estimated percent fan speed and by the rear hitch position. This seems quite obvious conclusions, but as said by Granger himself in his Nobel lecture, "Of course, many ridiculous papers appeared", referring to the use of Granger causality test for fields different from economics. Nevertheless, it remains a common method for time series causality analysis.

Chapter 5

Usage patterns identification

As previously introduced, the expected outcome of this thesis is to identify different usage patterns in data to support further analysis and maintenance. This can be reformulated as a classical unsupervised clustering task. Indeed, cluster analysis consists in dividing a set of items into *homogeneous* and well separated groups or *clusters* [28].
Hence, in the first part of the chapter, the main principles about clustering are presented. In particular, the K-Means algorithm and the silhouette analysis for clustering evaluation are described. Then, two different approaches for usage patterns identification are presented. The first one exploits the data synchronization performed in previous sections, considering each line of the dataset as an item to be clusterized. However, this approach does not take into account the time relationship between observations, treating each data point in isolation. The second method, instead, has the main advantage of considering the signals in their original sampling rates, avoiding time synchronization. It involves time series segmentation into subseries and the application of cluster analysis on suitable features extracted from each segment.

5.1 Clustering fundamentals

Cluster analysis is one of the most widely used techniques for data exploration [1]. It consists in grouping similar entities together in such a way that the similarity within data in the same cluster and the dissimilarity between points belonging to different groups are maximized.

If $C_1, C_2, \ldots, C_K$ denote clusters, for most of applications the following properties are required:

- Clustering is *complete*, in the sense that each observation belongs to at least one cluster:
$$C_1 \cup C_2 \cup \ldots \cup C_K = \{x_1, x_2, \ldots, x_n\} \tag{5.1}$$
If property (5.1) does not hold, clustering is said *partial*.

- Clustering is *exclusive*, in the sense that each observation belongs to no more than one cluster
$$C_j \cap C_i = \emptyset \ \forall j \neq i \tag{5.2}$$
If property (5.2) does not hold, clustering is said *non-exclusive*.

  If the hypothesis of exclusive clusters holds, partitional clustering algorithms are used.

The partitional algorithm selected for clustering is K-Means: it is fast, robust and simple. The main drawback of this method is that it can not perform well if data contains too much outliers or noisy points, if the expected clusters are overlapping too much or if they are characterized by not globular shapes or different densities [29]. However, in this application data are not expected to be particularly noisy, since they have been downsampled in the first part of the work. In addition, since the states are expected to be well separated, globular shapes hypothesis seems to be correct. It has been selected since it is easy to perform and leads to good results in many applications [30]. Another possible choice, not used in this thesis work, is agglomerative hierarchical clustering: it has the advantage of being able of recovering a hierarchy in data (if any) and does not require the user to provide the number of clusters as input to the algorithm [31]. However, in this application there is no particular evidence to support the hypothesis of any hierarchy in data.

Other algorithms commonly used are K-medoids and DBScan. With respect to K-means, the former is more robust to outliers but is more expensive [29]. The latter is a density-based clustering technique able to handle non globular shapes and different density clusters, but once again is more complex and requires the tuning of two hyperparameters [28].

Hence, despite its simplicity, K-Means is considered one of the universal clustering algorithm that can be applied to large dataset, can be parallelized and distributed and can be used in several contexts and real applications [30]. For these reasons, K-Means has been employed in this applications. However, as future work, the choice of the employed algorithm can be optimized, both analysing and validating results with domain experts, both employing R built-in function available in the library clValid [1] package for simultaneously clustering data and validating results by means of different internal indexes, in a sort of brute force approach.

[1] clValid

K-Means is a well-known partitioning method that associates each cluster with a centroid [1]. Then, it assigns each point to the cluster with the closest centroid according to the distance specified by the user as a parameter of the algorithm. In the following, the Euclidean distance have been used. The algorithm, described in Algorithm 1, is quite simple but requires to know the number k of clusters a priori: however, in common applications, it is not always known. It is possible to tune the algorithm maximizing several quality measures, among which the most popular one is silhouette score, introduced in the following.

The k-Means algorithm starts choosing randomly k points as centroids: since it is an iterative procedure, it is quite sensitive to initial starting conditions, meaning that produced clusters may vary from one run to another. Then, at each iteration it assigns points to the cluster with closer centroids and recomputes centroids (typically as the mean of the points in the cluster) in order to update them on the base of the new configuration of points. This procedure is ideally repeated until centroids do not change anymore from one iteration to the next one, that means convergence. It can be proved that this situation is reached after a finite number of iterations [30]. However, in common applications it is sufficient to stop the algorithm when it is close to convergence, forcing it if the centroids variations from an iteration to the next one is under a certain tolerance threshold. In this way, the algorithm usually stops after few iterations, since most of convergence happens in the first ones [30]. However, the obtained result is not necessarily a global optimum, but it could be a local one since it depends on the centroids initialization [30].

Algorithm 1 K-Means algorithm pseudo-code.
Input: Number of clusters K and dataset
Output: K complete and exclusive clusters

```
1: procedure K-Means(k, dataset)
2:     Randomly select k points as initial centroids.
3:     repeat:
4:         Form k cluster by assigning all points to the closest centroids
5:         Recompute the centroid of each cluster
6:     until the centroids don't change
       return Clusters C_1, C_2, ..., C_k
7: end procedure
```

K-Means algorithm is an heuristic approach to minimize intra-cluster distances, while it maximizes inter cluster distances: it is an optimization problem whose objective function or *clustering criterion*

$$\begin{aligned} F &: \mathcal{P}_K(\Omega) \longrightarrow \mathbf{R} \\ \{C_1, C_2, \ldots, C_k\} &\longrightarrow F(\{C_1, C_2, \ldots, C_k\}) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_i - \overline{x}_i||_2^2 \end{aligned} \tag{5.3}$$

is called *square error criterion*, where $\mathcal{P}_K(\Omega)$ denotes the set of all the possible partitions of the input dataset into K non-empty clusters, $x_{i,j}$ is the j-th observations of the i-th cluster and $\overline{x_i} = \frac{1}{|C_i|} \sum\limits_{x_j \in C_i}^{k_i} x_i \ \forall i = 1, \ldots, k$ is the centroid of the i-th cluster [1].

As previously introduced, one of the main drawbacks of the described algorithm is that the number of clusters need to be known. In common applications, since clustering is often used for understanding data in a unsupervised context, the number of clusters is unknown. In these cases, silhouette analysis can be used to measure the goodness of a clustering structure without any external information. To compute its value, after that data are grouped into k clusters, for each item x_i the following quantity are computed:

- the average dissimilarity $a(i)$ of x_i with all other data within the same cluster: the smaller is this value, the better is the assignment of x_i to its cluster. For item i in cluster C_h it is defined as

$$a(i) = \frac{1}{|C_h| - 1} \sum_{j \in C_h, i \neq j} ||x_i - x_j||_2$$

- the lowest average dissimilarity $b(i)$ of x_i to any other cluster of which x_i is not a member. For item i in cluster C_h it can be computed as

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} ||x_i - x_j||_2$$

  The second nearest cluster of item x_i,

$$j^* = \arg\min_{\mathrm{j} \neq \mathrm{i}} \left(\frac{1}{|\mathrm{C_k}|} \sum_{\mathrm{j} \in \mathrm{C_k}} ||\mathrm{x_i} - \mathrm{x_j}||_2 \right)$$

  is called *neighbouring cluster*.

Then the silhouette of x_i belonging to cluster C_h is defined as

$$s(i) = \begin{cases} \dfrac{b(i) - a(i)}{\max\{a(i), b(i)\}}, & \text{if } |\mathrm{C_h}| > 1 \\ 0, & \text{if } |\mathrm{C_h}| = 1 \end{cases}$$

From definition, it follows that the silhouette takes values in the interval $[-1,1]$. When the silhouette takes values close to 1, it means that the cluster is well separated. On the other hand, values close to 0 describe the situation in which the given point is characterized by a tiny margin from the decision boundary between the cluster to which it is assigned and another one. Finally, a negative silhouette value indicates instead that the point might have been wrongly assigned.

Computing the average of $s(i)$ over all available data points it is possible to obtain a measure of how closely clustered the data in the same group are, while the overall average of $s(i)$ gives a measure of how properly data have been grouped considering the entire set of data points.

5.2 Clustering by value

The first method applied to identify vehicle's states is clustering by value. Since data have been synchronized, it is possible to consider each record of the dataset as a point for clustering. This method focuses on the value of the single points instead of considering the general trend since each point is taken in isolation as an item to cluster. The main advantages of this method are that it is a classical approach, easy to be performed. On the other hand, it does not take into account the temporal order of observations.

Feature selection is performed coherently with Chapter 4 conclusions: neglecting strongly correlated features and not informative SPNs, the parameters considered in the analysis are the engine coolant temperature (SPN 110), the engine speed (SPN 190), the engine fuel delivery pressure (SPN 94), the transmission selected gear (SPN 524) and the charge pressure (SPN 30789).

Since the dimensionality of the dataset is limited, no dimensionality reduction is needed. On the other hand, since K-Means involves distances, in order to make comparable different quantities, the dataset is standardized, applying to each signal the Z-score normalization:

$$z(t_k) = \frac{x(t_k) - \mu_x}{\sigma_x}$$

Then, K-Means algorithm can be applied to the obtained dataset. In particular, silhouette analysis is performed to identify the optimal number of clusters: hence the k-Means algorithm is run for a number of clusters varying from 2 to 9 and

the best value is the one that maximizes the silhouette score. Silhouette scores obtained for each value of k, summarized in Table 5.1, are graphically displayed in Figure 5.1.

Number of Clusters	Silhouette score
2	0.786
3	0.742
4	0.761
5	0.769
6	0.7745
7	0.775
8	0.656
9	0.653

Table 5.1: Silhouette score for each number of clusters obtained performing clustering by values, varying the number of clusters from 2 to 9.

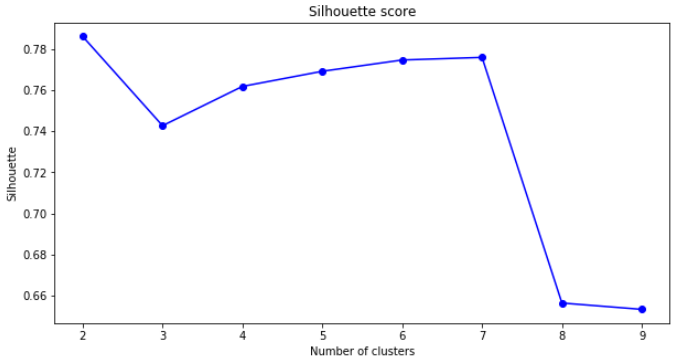

Figure 5.1: Silhouette values for each number of clusters obtained performing clustering by values, varying the number of clusters from 2 to 9.

For $k = 2$ the silhouette score reaches the maximum value, close to 0.8, meaning that partitioning all data points in just two groups might be a right clustering. However, the silhouette score represents only quantitative measure for selecting the best number of clusters, but it could be a biased measure if used to validate obtained results. Since this application is an unsupervised problem, no additional information about the ground truth values is provided. In this cases, the final cluster results should be validated by a domain expert or by qualitative considerations about parameters distribution, separately in each cluster.

Finally, applying the K-Means algorithm with $k = 2$, data points are divided into two groups, whose number of records are summarized in Table 5.2.

Cluster	Number of records
1	255036
2	39229

Table 5.2: Number of records in each cluster obtained performing clustering by values applying K-Means algorithm with $k = 2$

To interpret cluster results, some statistics are computed for each SPN in each cluster.

As it can be graphically seen from Figure 5.2, SPN 190 is the more discriminant factor among clusters. Indeed, low values of engine speed are associated with cluster 1, while higher values are associated with cluster 2. The same structure holds also for the engine load: this was trivially predictable thanks to the strong positive correlation between SPN 190 and 30000. For the same reasoning, the same conclusions hold for spn 32061 and 183.
The same general behaviour, even if with less precise boundaries, can be seen for the engine fuel delivery pressure (SPN 94). For the remaining SPNs, no particular differences in distributions can be noticed. The useful quantities to summarize the signals behaviour within each clusters are reported in Table 5.3.

The highlighted differences of SPNs distributions among different clusters can be seen thanks to violin plots, shown in Figure 5.3.

In conclusion, it is possible to say that cluster 2 contains observations characterized by extremer values, indicating a higher workload, while cluster 1 contains observations that are more likely to be collected during an idle phase or standard working phases, without any sudden change in the vehicle behaviour. According to the number of items in each cluster, it can be noticed that the vehicle is mainly in idle or standard working state: it is coherent with the expected results, since the vehicle under analysis is mainly used for testing devices and driver assistance systems, as previously introduced. The result of this procedure is then to automatically identify some thresholds, as the manually set ones used by the company, to identify the vehicle's states. However, it is not possible to make a comparison with the ones actually employed by the company since they are mainly based on SPNs not available for this application and other additional information provided by digital input reports.

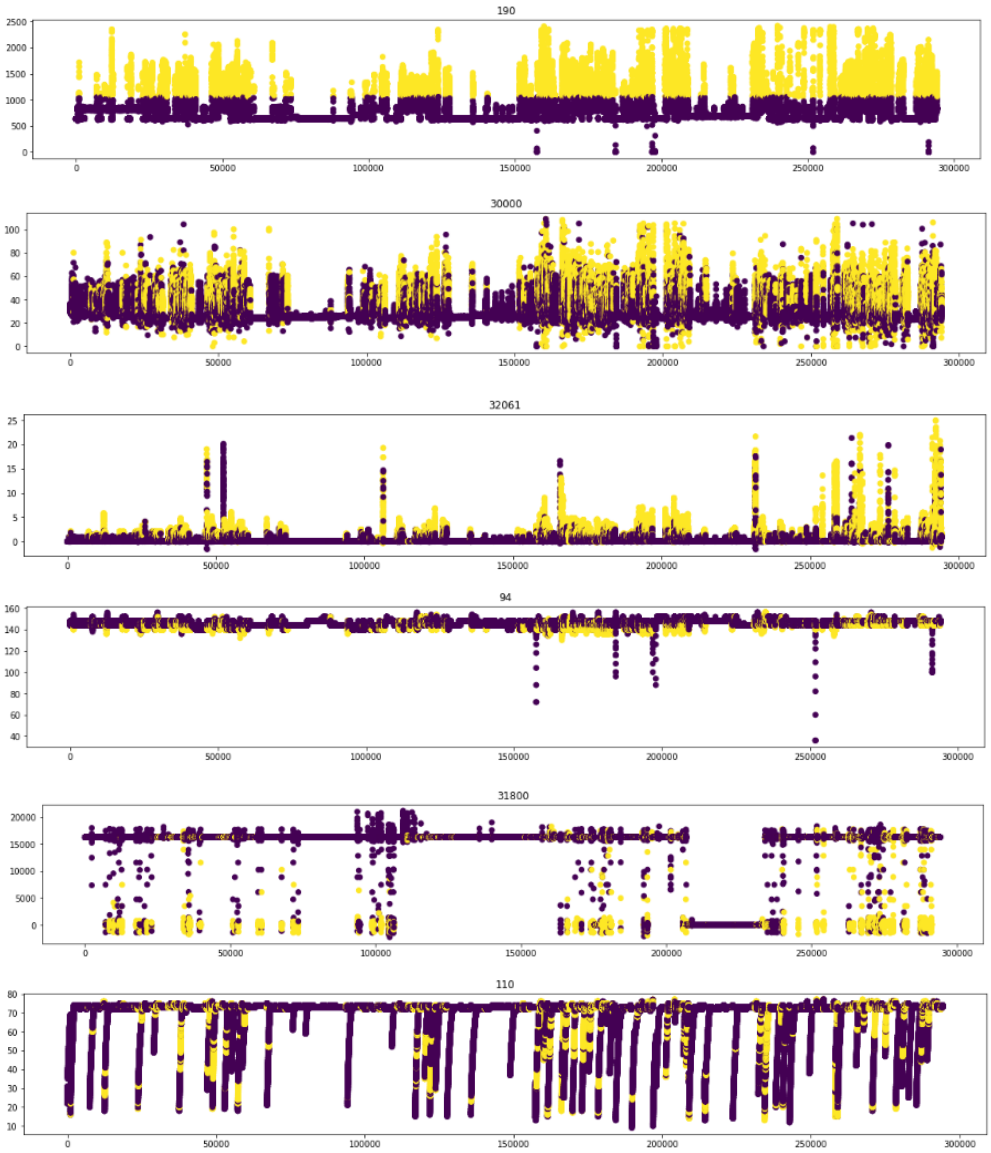

Figure 5.2: Time series plot, highlighting clusters, for representative SPNs Data points belonging to cluster 1 are displayed in violet, while points belonging to cluster 2 are the yellow ones.

SPN	Cluster	mean	std	min	25%	50%	75%	max
	1	145.9	2.4	36.0	144.0	146.0	148.0	156.0
94	2	144.9	3.2	130.0	144.0	144.0	148.0	156.0
	1	67.9	11.3	9.0	72.0	73.0	73.0	77.0
110	2	70.0	9.9	13.0	72.0	73.0	74.0	77.0
	1	11488.9	122.7	11320.5	11396.5	11457.5	11547.5	11751.5
182	2	11530.5	141.0	11321.5	11413.0	11500.5	11714.0	11751.5
	1	3.7	0.9	0.0	3.2	3.4	3.8	18.8
183	2	9.5	3.5	0.0	7.1	8.9	10.9	38.7
	1	722.2	100.2	0.0	637.3	666.0	830.5	1061.6
190	2	1400.8	259.3	1061.5	1201.8	1350.1	1533.6	2422.9
	1	1	0.1	1.0	1.0	1.0	1.0	5.0
524	2	2.0	1.36	1.0	1.0	1.0	1.0	3.0
	1	100.0	0.4	67.6	100.0	100.0	100.0	100.0
975	2	100.0	0.4	69.2	100.0	100.0	100.0	100.0
	1	28.5	6.2	0.0	24.8	25.9	28.9	109.1
30000	2	43.1	11.1	0.0	35.9	41.1	48.4	109.2
	1	89.7	3.1	-16.8	88.1	88.2	89.4	106.9
30694	2	90.8	3.6	87.4	88.2	88.6	96.1	96.9
	1	205.3	16.3	-24.4	199.6	207.8	213.1	267.5
30789	2	208.1	11.1	19.0	202.9	208.0	216.3	263.7
	1	0.0	1.0	-8.7	0.0	0.0	0.0	93.8
31391	2	-0.0	0.0036	-0.6	0.0	0.0	0.0	0.3
	1	14207.4	5389.2	-2255.4	16260.7	16260.7	16260.7	21184.7
31800	2	10301.8	7808.2	-1928.2	64.3	16260.7	16260.7	18248.7
	1	0.2	0.5	-1.6	-5 e-46	1 e-136	1e-07	21.4
32061	2	1.9	1.9	-1.3	1.2	1.6	2.4	25.0

Table 5.3: Difference in SPNs distributions between the two clusters obtained performing clustering by values applying K-Means algorithm with $k = 2$
Mean is substituted by mode in case of SPN 524 which is a categorical variable

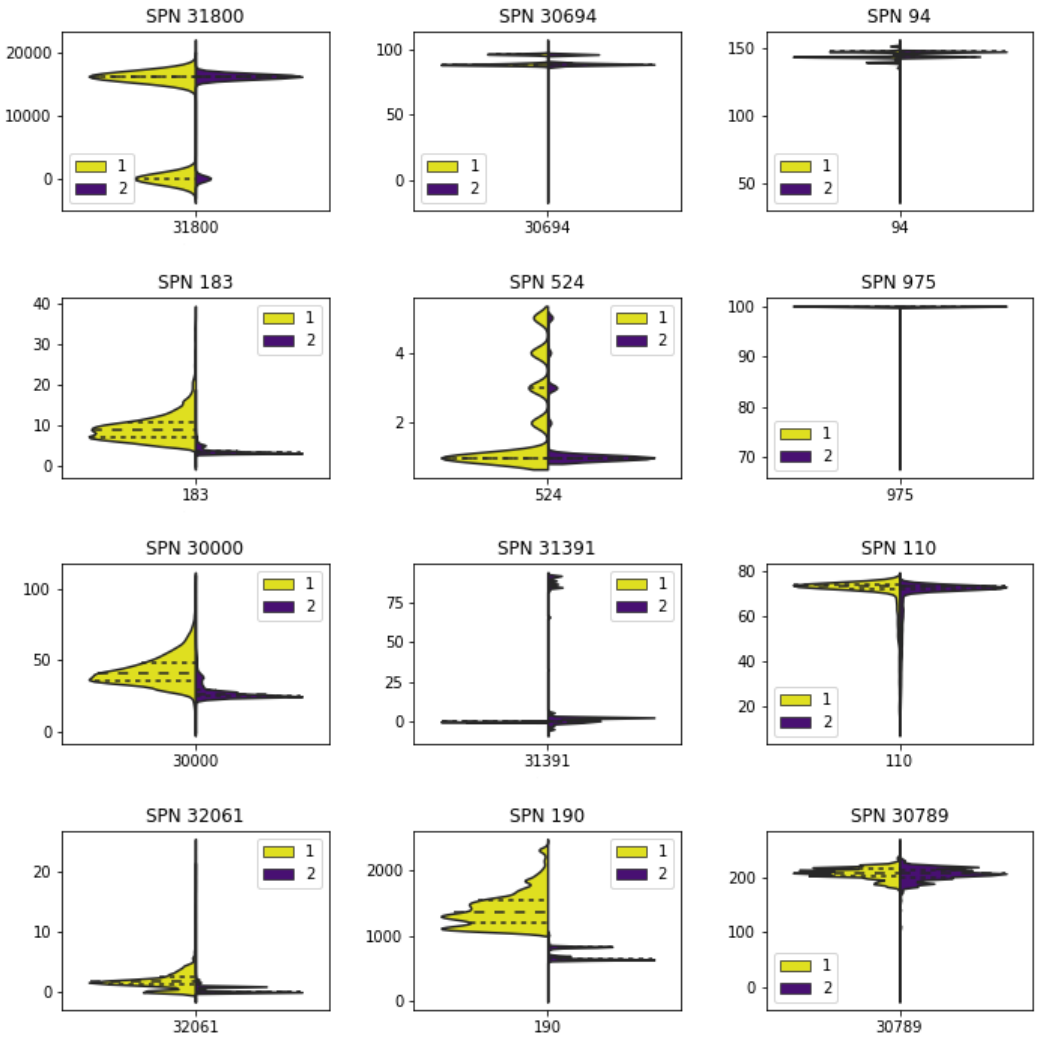

Figure 5.3: Violins plot to highlight the differences in SPNs distributions between the two clusters obtained performing clustering by values applying K-Means algorithm with $k = 2$

5.3 Clustering extracting features in frequency domain

As previously introduced, CAN data clustering is an unsupervised time series clustering task. It is particularly challenging, since the addition of time order among observations increases the dimensionality and the complexity of the problem under analysis. Indeed, in a time series clustering scenario, typically the number of features is extremely high, also due to the fact that the time dimension is taken into account in the analysis [32]. In addition, the temporal order which characterize the observations causes the features to be often mutually dependent or can introduce some hidden features that may not be directly inspectable [32]. Finally, this kind of data are unlikely to be associated with a label. Hence, in a time series clustering problem is important to reduce dimensionality, extract significant features from data and discover the hidden relationships between points [32]. In addition, as highlighted in previous section, because of the high complexity, a cluster model based only on data point values that does not take into account the temporal relationship between items may be not sufficient.

There exist many different clustering algorithms and techniques for problems involving time series, but most of them are not suitable for this kind of application since they are tools developed for univariate time series clustering or for supervised context [7].

A possible way to overcome this issue consists in computing different features based on the series distribution and shape and use those as input variables for clustering [7]. This approach is presented in the following: as first step, it requires to break the time series into segments of a fixed length to be clustered. Since the optimal length of segments is unknown and there exists no multivariate algorithm to infer it, VALMOD algorithm [33], a scalable one dimensional approach for discovering repeated patterns of variable length in data, is applied. The VALMOD output will be, for each signal, a list of pairs associated with different lengths, ranked by closeness of the two considered segments. Hence, taking into account only the first distances for each signal and combining together the obtained results, it is possible to find an approximation of the optimal segment length.
Then, features in the frequency domain are extracted from each segment: this procedure allows to characterize the general shape of each window in terms of its variations and their respective rapidity and amplitudes by inspecting the signals spectral content.
The main advantage of this method is that there is no need to synchronize data, since each feature is extracted separately for each dimension. However, to limit the computational time required by VALMOD algorithm for finding repeated patterns in ultrafine data when the range of variability of patterns length is pretty wide,

data resampled at $1Hz$ are considered to determine the optimal segments length, obtaining an approximation of repeated patterns for original data.

5.3.1 Time series segmentation: an application of VALMOD algorithm

The problem of time series segmentation can be formulated as it follows [23]: find a partitioning of a given time series $X(t)$ into s internally homogeneous segments.

The searched segments are often characterized by the same length, even if there exist some techniques for variable-length segmentation: one of them is based on applying a cluster algorithm with the constraint of grouping in the same cluster observations that are sequentially taken over time [23]. However, because of the high complexity of the problem, for the sake of simplicity fixed length segmentation is applied in the following.

In a multivariate context, several variables are taken into account and the segmentation of each component need to be synchronized. Of course, because of the hidden underlying model of the considered system, components are correlated. If combining results obtained for few significant signals is not sufficient for correct segmentation of the entire multivariate time series, what is typically done is monitor only some principal components [34]. However, for the following analysis, it is not necessary since satisfactory results are obtained without employing principal components.

In order to find the optimal length of segments, the VALMOD algorithm is applied separately to each significant signal. Indeed, it is a tool for discovering repeated patterns in data without requiring the user to provide their length, unknown in most of applications even to data domain experts. The algorithm proposed in [33], given a univariate time series $X(t)$, finds the subsequence pairs with the smallest Euclidean distance of each length in the user provided range $[L, U]$. Formally, a subsequence is defined as it follows [33]:

Definition 5.3.1 *Given a time series $X \in \mathbf{R}^n$, a* subsequence *$X_{i,l} \in \mathbf{R}^l$ of X is the subset of l sequential values of X starting from position i.*

The subsequence pairs with smallest Euclidean distance are called *motifs* [33]:

Definition 5.3.2 *Given two subsequences $X_{a,l}$, $X_{b,l}$ of length l of the same time series $X \in \mathbf{R}^n$, $X_{a,l}$ and $X_{b,l}$ are called* motif pair *if and only if*

$$dist(X_{a,l}, X_{b,l}) \leq dist(X_{i,l}, X_{j,l}) \ \forall i, j \in [1,2,\ldots,n-l+1],\ a \neq b\ i \neq j.$$

The distances between a subsequence with all the other subsequences from the same series are stored inside an ordered array, the *distance profile*. From it, it

is possible to compute the *matrix profile* $MP \in \mathbf{R}^{n-l+1}$ evaluating the minimum value of every distance profile vector. The matrix profile contains the so called *exclusion-zone*, heuristically set to $\frac{l}{2}$, to avoid trivial matches of a pattern with itself or with a largely overlapping one [33].

The computation of the matrix profile for the smallest subsequence length L is the first step of VALMOD algorithm: indeed, it minimizes computational cost exploiting the property that if the nearest neighbor of $X_{i,L}$ is $X_{j,L}$, then probably the nearest neighbor of $X_{i,L+1}$ will be $X_{j,L+1}$. But since this property holds only with few lags shift, a new vector called *lower bound distance profile* is introduced, containing the lower bound distance between $X_{i,l+k}$ and $X_{j,l+k}$, $\forall k \in [1,2,\ldots]$ that can help to prune the number of needed computations [33]: indeed, if a best-so-far pair motifs is characterized by a distance d, it is sufficient to calculate the exact distance only for segments with lower bound strictly smaller than d. In addition, a maximum number p of segments taken into account is fixed: hence, it is sufficient to check the inequality only for the first smaller p lower bounds. In the worst case, only $\mathcal{O}(np)$ exact length at each iterations are computed [33].

The first step to evaluate the lower bound distance profile is defining the lower bounding Euclidean distance: supposing to know the z-normalized Euclidean distance $d^l_{i,j}$ of two sequences of length l $X_{i,l}$ and $X_{j,l}$, it is possible to estimate the distance $d^{l+k}_{i,j}$ between the subsequences $X_{i,l+k}$ and $X_{j,l+k}$ finding a lower bound function $LB(d)$ such that $LB((d_{i,j})^{l+k}) \leq d^{l+k}_{i,j}$, as shown in [33]. As highlighted in the reference article, the main problem in estimation of z-normalized Euclidean distance is that the mean and the standard variations change as the length of the given sequence increases, hence $\mu_{i,l+k}$ and $\sigma_{i,l+k}$ are unknown and treated as variables [33]:

$$
\begin{aligned}
d^{l+k}_{i,j} &\geq \min_{\mu_{i,l+k},\sigma_{i,l+k}} \sqrt{\sum_{n=1}^{l} \left(\frac{X(i+n-1)-\mu_{i,l+k}}{\sigma_{i,l+k}} - \frac{X(j+n-1)-\mu_{j,l+k}}{\sigma_{j,l+k}} \right)^2} \\
&= \min_{\mu_{i,l+k},\sigma_{i,l+k}} \frac{\sigma_{j,l}}{\sigma_{j,l+k}} \sqrt{\sum_{n=1}^{l} \left(\frac{X(i+n-1)-\mu_{i,l+k}}{\frac{\sigma_{i,l+k}\sigma_{j,l}}{\sigma_{j,l+k}}} - \frac{X(j+n-1)-\mu_{j,l+k}}{\sigma_{j,l}} \right)^2} \\
&= \min_{\mu'\sigma'} \frac{\sigma_{j,l}}{\sigma_{j,l+k}} \sqrt{\sum_{n=1}^{l} \left(\frac{X(i+n-1)-\mu'}{\sigma'} - \frac{X(j+n-1)-\mu_{j,l}}{\sigma_{j,l}} \right)^2}
\end{aligned}
$$

Since previous expression defines an unconstrained optimization problem whose objective function is convex, it is possible to obtain the minimum value, that is $LB(d^{l+k}_{i,j})$, by solving the first order optimality conditions

$$
\frac{\partial LB(d^{l+k}_{i,j})}{\partial \mu'} = 0 \qquad \text{and} \qquad \frac{\partial \mathrm{LB}(\mathrm{d}^{\mathrm{l+k}}_{\mathrm{i,j}})}{\partial \sigma'} = 0
$$

Hence the lower bound function $LB(d_{i,j}^{l+k})$ for the z-normalized Euclidean distance between the considered segments can be written as [33]

$$LB(d_{i,j}^{l+k}) = \begin{cases} \sqrt{l}\dfrac{\sigma_{j,l}}{\sigma_{j,l+k}} & \text{if } \mathrm{q_{i,j}} \leq 0 \\ \sqrt{l(1-q_{i,j}^2)}\dfrac{\sigma_{j,l}}{\sigma_{j,l+k}} & \text{otherwise} \end{cases}$$

where

$$q_{i,j} = \frac{\sum_{n=1}^{l} \frac{X(j+n)X(i+n-1)}{l} - \mu_{i,l}\mu_{j,l}}{\sigma_{i,l}\sigma_{j,l}}$$

By evaluating the previous expression with all possible subsequences with length $l+k$ and sorting results in ascending order, it is possible to speed up computations, pruning the number of operations according to the previous described technique.

The implementation [2] provided by the authors of [33] has been adopted, obtaining as final output a vector of variable-length matrix profile VALMP containing the k closest motif pairs. It has the following structure:

- i-th sequence and its nearest neighbor
- their straight Euclidean distance
- their length-normalized Euclidean distance, necessary for ranking different length motifs
- the motif lengths

The idea is to apply the described algorithm to the most significant SPNs in order to find the lengths of the closest motifs and combine together the obtained results for different SPNs to determine an approximation of the optimal segment length for the multivariate time series.
The searched motif lengths range was determined thanks to domain experts, keeping in mind that a too tiny time window would be not enough informative to support the analysis, whereas if it is too wide it would be hard to identify usage patterns. In other words, the segments obtained dividing the original series according to a suitable length should be characterized by an homogeneous and recognizable usage pattern. Hence the searched motif length ranges from 2 to 10 minutes. Since it is a pretty wide range, data resampled at $1Hz$ are used to limit the computational time.

[2] helios.mi.parisdescartes.fr/mlinardi/VALMOD

As first trial, given the results of clustering by value, VALMOD is applied only to the engine speed signal (SPN 190). Results are summarized in table 5.4: as it is possible to see, the best length for SPN 190 is 2 minutes. Then, VALMOD has been applied to the SPNs that are not correlated with 190 and that are considered significant for the analysis: however, for SPN 110 and 30789 no motifs sufficiently close were found in the given range. Furthermore, since the goal of this analysis is to inspect the spectral content of each signal, the transmission selected gear (SPN 524) is not considered because it is a quite constant SPN and then not suitable for spectral analysis since all the information will be focus on lower frequencies. Finally, the fuel delivery pressure (SPN 94) gives similar results to the ones obtained for SPN 190, hence 2 minutes was selected as window size. Working cycles smaller than 2 minutes were removed, losing about 10 minutes of data.

As result of the described steps, 2405 segments are identified.

Offset1	Offset2	Normalized distance	motif length (s)
950	94087	0.049898	120
9787	49225	0.051419	120
9788	49226	0.052060	120
9789	49227	0.052633	120
9790	49228	0.052927	120
9791	49229	0.053718	120
9792	49230	0.054584	120
9793	49231	0.054954	120
9794	49232	0.056010	120
9795	49233	0.057158	120
9678	30491	0.057262	120
9797	49235	0.057567	120
949	94086	0.058008	120
9796	49234	0.058313	120
9677	30490	0.058516	120
9798	49236	0.058993	120
9799	49237	0.060244	120
9694	22728	0.060320	120
9676	30489	0.060781	120
9804	120205	0.060918	120
9693	22727	0.061090	120
9786	49224	0.061358	120
9800	49238	0.061556	120
9692	22726	0.061773	120
9675	30488	0.061905	120
9691	22725	0.062418	120
9690	22724	0.063302	120
2358	97378	0.063559	120
2359	97379	0.063658	120
9805	120206	0.063910	120
9801	49239	0.064137	120
9689	22723	0.064292	120
2360	97380	0.064437	120
2357	97377	0.064486	120
7015	122379	0.064636	120
7016	122380	0.064644	120
9674	30487	0.064760	120
7014	122378	0.064795	120
2355	97375	0.064932	120
9688	22722	0.065310	120
7013	122377	0.065588	120

Table 5.4: Top 40 motifs obtained as output of VALMOD algorithm applied to SPN 190 with sampling rate $1Hz$

5.3.2 Feature extraction

As introduced in previous sections, applying a Fourier Transform can be useful to highlight, in the frequency domain, properties that would remain undisclosed in the temporal domain. Indeed, a way to identify usage patterns is describing the frequencies and the amplitude of signals variations in different segments and Fourier coefficients describe how the signal is distributed over the frequencies. As first step, SPNs with informative spectral content must be selected. In order to see the spectral content of each SPN, the biggest working cycle was selected, corresponding to almost 4 hours of work. For each segment in this cycle and for each SPN, the spectrum (shown for significant segments and SPNs in Figure 5.4) is inspected in order to identify SPNs with enough spectral content: they corresponds to signals with enough variations within the time window of 2 minutes, such that some aggregate values can be computed separately in their low, medium and high frequencies. The spectral of signals characterized by slower variations instead shows all its information in correspondence of frequencies close to 0: applying frequencies analysis in this case is almost useless since the zero frequency, also known as DC component, is equal to the temporal average value of the signal [11]. Indeed, the information corresponding to zero frequency is commonly not interesting for what concerns spectrum analysis and therefore the mean value of the signal should be subtracted before applying the Fourier transform [35].

There are two main reasons for which a signal is not suitable for analysis in frequency domain, described in the following. The first one is the nature of the signal: it is the case of transmission selected gear (SPN 524), of the estimated percent fan speed (SPN 975), of the digging depth (SPN 31800) or of the amount of particulate matter C method (SPN 31391). These signals, characterized by a more constant behaviour due to the nature of the monitored parameters, should not be analysed by means of frequencies analysis. The second reason is that the sampling rate is not appropriate for describing the signal variations: this is the case of engine coolant temperature (SPN 110, $1Hz$), fuel tank level (SPN 32061, $9.6Hz$), engine fuel delivery pressure (SPN 94, $2Hz$), engine trip fuel (SPN 182, $1Hz$). For these signals, a higher sampling rate could be a correct choice, even if the measurement instrument should have an adequate sensitivity. Finally, SPNs with proper spectral content are the engine speed (SPN 190, $50Hz$), engine fuel rate (SPN 183, $10Hz$), engine load (SPN 30000, $20Hz$), charge pressure (SPN 30789, $33.33Hz$) and rear hitch position (SPN 30694, $10Hz$). Based on the conclusions drawn in the previous analyses, just one parameter is considered in the following, the engine speed.
However, the following procedure is general and can be easily extended to multivariate clustering analysis if data with suitable form and spectral content are provided.

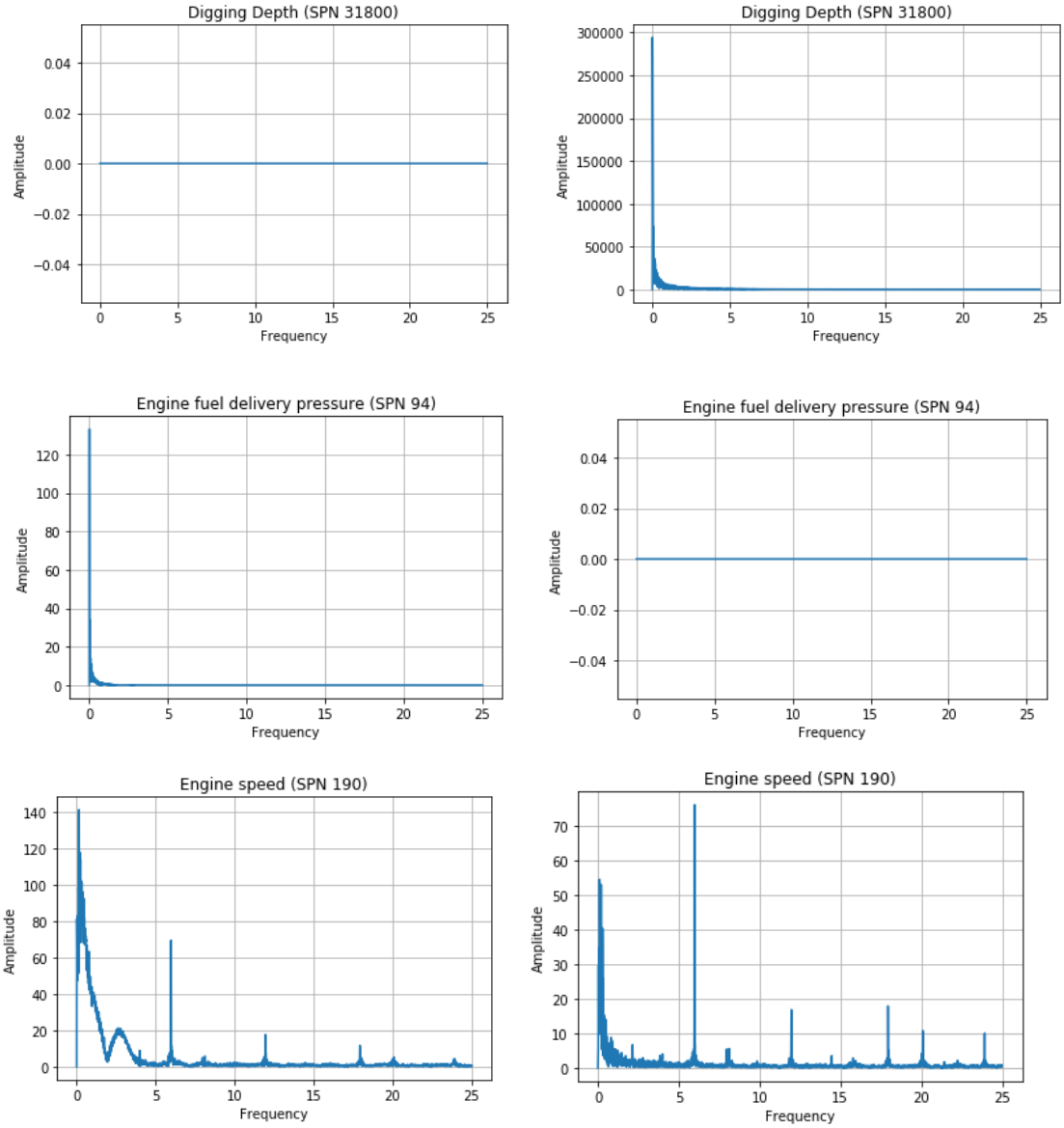

Figure 5.4: Spectrum in two different segments for SPN 31800, 94 and 190

According to frequencies distribution for each segment of the considered working cycle as the two reported in Figure 5.4, the frequencies for SPN 190 are said to be *low* from $0Hz$ to $5Hz$, *medium* from $5Hz$ to $15Hz$ and *high* from $15Hz$ to $25Hz$.

Then, the same procedure has been performed on the longest cycle as trial is applied to each cycle characterized by a time length greater than 2 minutes: in each segment the Fourier transform of the signal is computed and only positive coefficients are considered exploiting the symmetry of Fourier coefficients for real

signals. Then, for each subband, the power spectrum value [36] is computed: given a discrete real-valued signal $\{x_n\}$, it is defined as the square of the signal quadratic norm [11]. If $(\alpha_0, \alpha_1, \ldots, \alpha_N)$ represent the signal discrete-time Fourier positive coefficients, thanks to Parseval equality, its power spectrum value can be computed as

$$||x_2||_2^2 = \sum_{i=0}^{n} |\alpha_n|^2$$

The power value for low, medium and high frequencies is denoted rispectively as *power_lf*, *power_mf* and *power_hf*. Then, to describe peaks values and their frequencies, the maximum absolute value of Fourier coefficients and the corresponding frequency in each subband are computed, namely *peak_lf*, *peak_mf*, *peak_hf*, *peakfreq_lf*, *peakfreq_mf* and *peakfreq_hf*. Finally, the last feature considered is the temporal mean of the signal, *Tmean*.

The distributions of the computed features are summarized in Figure 5.5.

	power_lf	power_mf	power_hf	peak_lf	peak_mf	peak_hf	peakfreq_lf	peakfreq_mf	peakfreq_hf	Tmean
count	2.405000e+03	2.405000e+03	2405.000000	2.405000e+03	2405.000000	2405.000000	2405.000000	2405.000000	2405.000000	2405.000000
mean	1.454371e+08	5.331263e+04	12740.583304	8.153570e+07	3051.716480	101.232225	4.991701	9.991704	9.991555	817.819308
std	3.704597e+08	1.337574e+05	37645.246454	2.321410e+08	4332.323323	104.154228	0.000303	0.000367	0.000908	224.109650
min	1.046584e+04	6.700537e+02	105.185929	5.802383e+02	33.970388	2.105578	4.991667	9.989977	9.979990	639.125330
25%	4.128434e+04	5.523561e+03	861.590276	4.822105e+03	514.203502	30.604956	4.991667	9.991667	9.991667	642.500242
50%	1.231099e+05	1.034579e+04	1398.089522	1.353370e+04	1280.426604	68.227544	4.991667	9.991667	9.991667	716.841552
75%	6.614540e+07	2.540666e+04	3327.019118	2.480437e+07	3662.034428	131.755858	4.991667	9.991667	9.991667	865.657611
max	3.666293e+09	1.704448e+06	413712.423675	2.595187e+09	34521.081614	788.614852	4.996663	9.996665	9.991667	1892.204270

Figure 5.5: Mean, standard deviation, minimum, quartiles and maximum of the new features

Studying the correlation between the extracted features, it is possible to notice that most of them are strongly correlated, as shown in Figure 5.6. For example, the power in low, medium and high frequencies and the peak value for low frequencies are characterized by high positive correlation, whereas the peak frequency in the high subband is strongly negatively correlated with both the peak frequency in the other subbands. Applying the same reasons as the ones employed in feature selection for the previous clustering strategy, only the power in low frequencies subband, the peak value for high frequencies and the mean of the signal over time are considered.

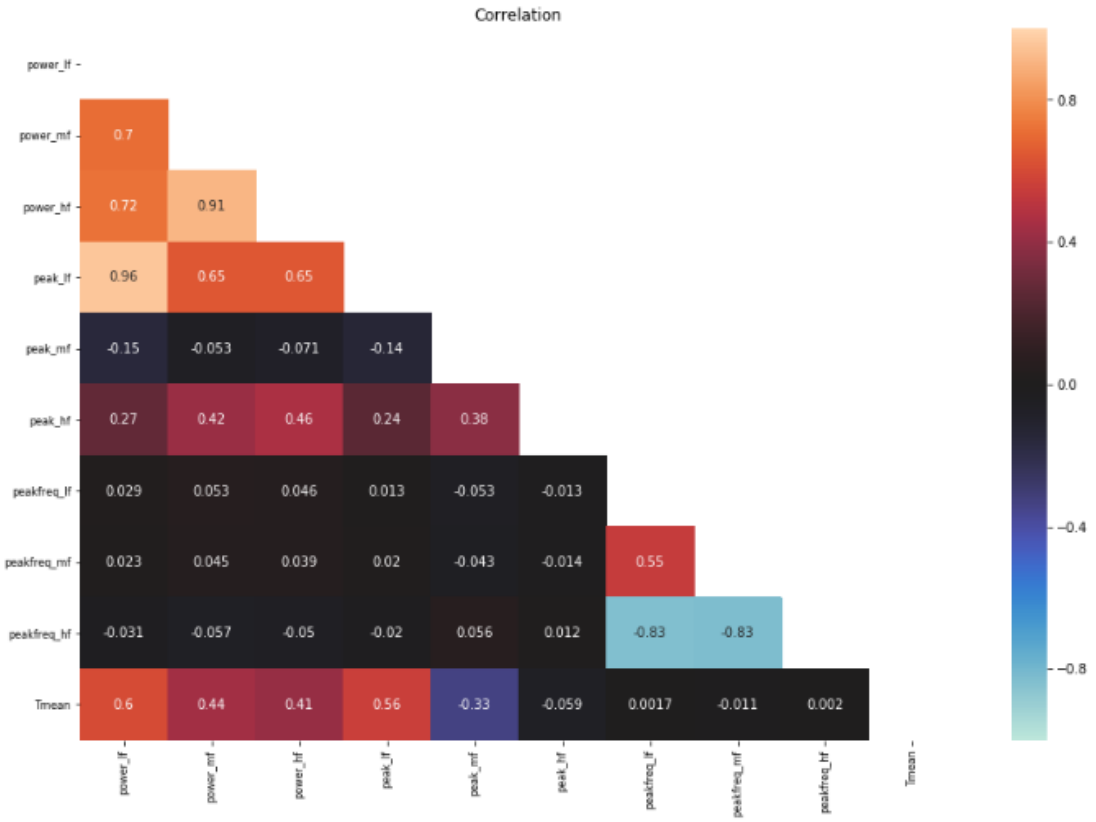

Figure 5.6: Correlation between new features extracted from engine speed (SPN 190)

5.3.3 Analysis of clustering outcomes

Once the previous described features have been computed for each segment, the K-Means algorithm is applied to the resulting dataset. In order to find the optimal number of clusters, silhouette analysis is again performed, by varying k from 2 to 19. Results, summarized in Table 5.5, are graphically displayed in Figure 5.7.

For $k = 2$ the silhouette score is close to 0.57, meaning that partitioning all data points in just two groups might be a right clustering. As in the previous case, silhouette score represents only a quantitative measure for evaluating unsupervised cluster results. They should be, however, validated by a domain expert or by considering other additional information: examples could be digital inputs report, such as data collected by embedded devices installed on the vehicle, or spatial information such as the ones collected by GPS to infer, as example, the speed of the vehicle [4]. Another option could be collecting few measurements in a supervised context and use their ground truth values to obtain a measure of how well the

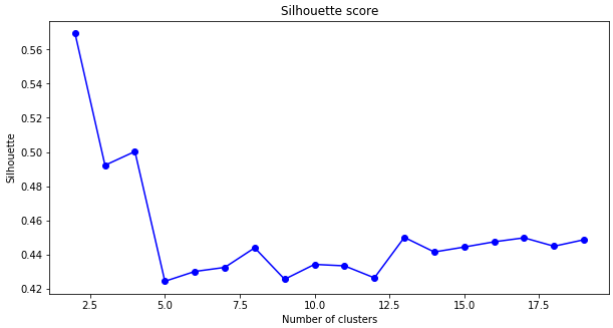

Figure 5.7: Silhouette score for each number of clusters obtained performing clustering by features, varying the number of clusters from 2 to 19.

Number of Clusters	Silhouette score
2	0.5693
3	0.4923
4	0.5003
5	0.4243
6	0.4301
7	0.4325
8	0.4440
9	0.4255
10	0.4343
11	0.4334
12	0.4264
13	0.4501
14	0.4415
15	0.4444
16	0.4475
17	0.4498
18	0.4449
19	0.4487

Table 5.5: Silhouette score for each number of clusters obtained performing clustering by features, varying the number of clusters from 2 to 19.

build model fit new data.

Applying the K-Means algorithm with $k = 2$, data points are divided into two groups, whose number of records are shown in Table 5.6.

Cluster	Number of records
1	2021
2	384

Table 5.6: Number of records in each cluster obtained performing clustering by features applying K-Means algorithm with $k = 2$.

By averaging the feature values separately for each cluster, summarized in Table 5.9, it is possible to explore clusters characteristics and pairwise similarity. Indeed, it can be seen that cluster 2 contains segments characterized by higher frequencies, hence higher and rapid variations in time domain are expected. On the other hand, cluster 1 is characterized by lower frequencies, corresponding to smaller and slower variations in time domain.

Cluster	power_lf	power_mf	power_hf	peak_lf	peak_mf	peak_hf	TMean
1	7.52 e+8	215282.6	56342.1	4.38e+8	939.0	122.9	1227.1
2	3.02e+7	22537.5	4456.1	1.4e+07	3453.1	97.1	740.1

Table 5.7: Mean value, computed separately for each cluster, of power and peaks in low, medium and high subbands and temporal mean value.

In addition, some representative segments for each cluster are shown in Figure 5.9. As it is possible to see, clusters are well separated according to the segments variations: indeed, cluster 1 (represented in red in Figure 5.9), corresponds to a working vehicle since it is characterized by a higher workload and a more aggressive driving style, probably with rapid acceleration and braking. On the other hand, cluster 2 (whose segments are represented in green and gray in Figure 5.9) is characterized by a more regular workload. It is possible to see the differences between segments assigned to the two different clusters in the frequency features distribution violin plots shown in Figure 5.8. Cluster 2 is characterized by lower powers both in low, medium and high frequencies and characterized by lower variations with respect to clusters 1. Furthermore, the same behaviour can be noticed in the distribution of peaks in low frequencies. On the other hand, there are no evident differences in peaks in high frequencies. Finally, cluster 1 is characterized by a higher temporal mean, while it is possible to see two peaks in the temporal mean distribution for segments belonging to cluster 2.

The presence of the two peaks can be explained considering that, according to domain experts, at least two states should be identified: moving and idle. As can be seen from Figure 5.9, the algorithm identified a higher workload and a more

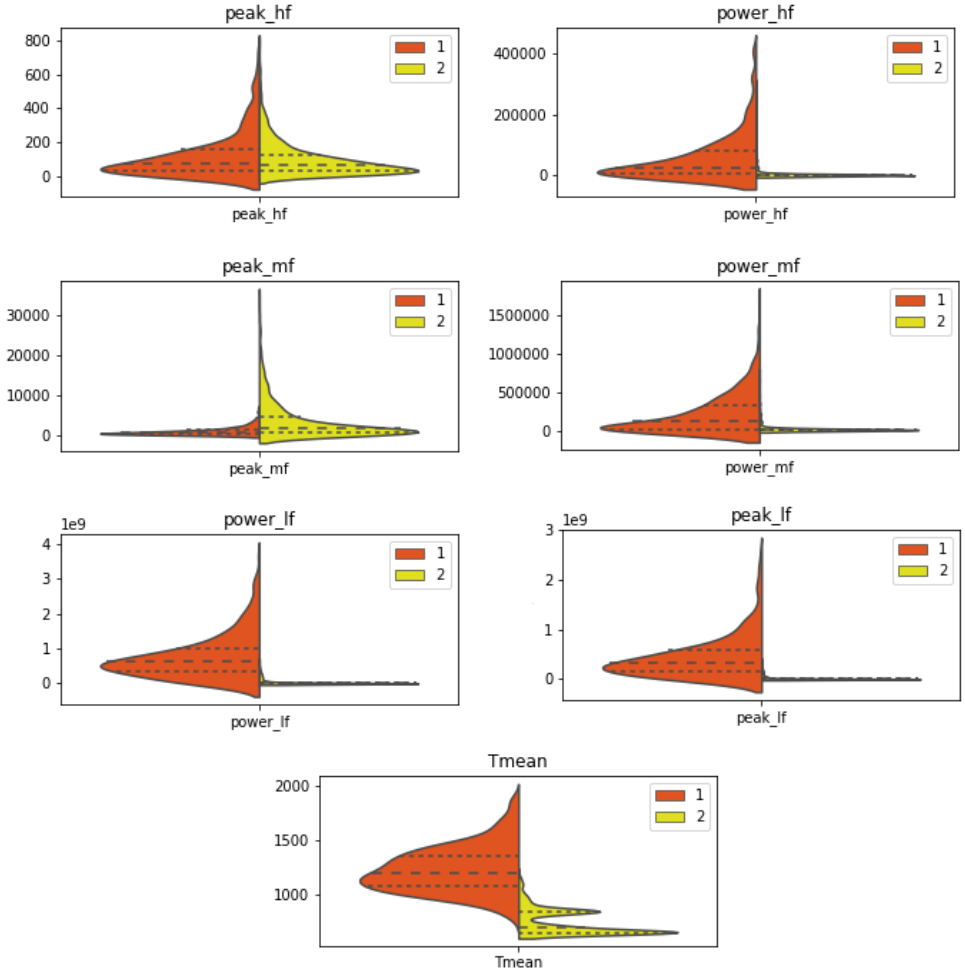

Figure 5.8: Violin plots showing the different distribution of frequency domain features between the 2 identified clusters

regular one, but it is not capable of telling an idle state (in gray) from a moving with regular driving style one (in green).

Indeed, since their variations are characterized by similar amplitudes and frequencies, spectral analysis is not an adequate tool to discern one from another. Indeed, the only visible difference between segments associated with this two usage patterns is the mean value of engine speed: moving vehicle is associated with higher engine speeds, whereas when the vehicle is in idle state, lower values are collected. Hence it would be possible to make a distinction between these states by means of suitable thresholds, as the ones identified as result from the previous clustering strategy: since the mean value of engine speed for cluster representing regular

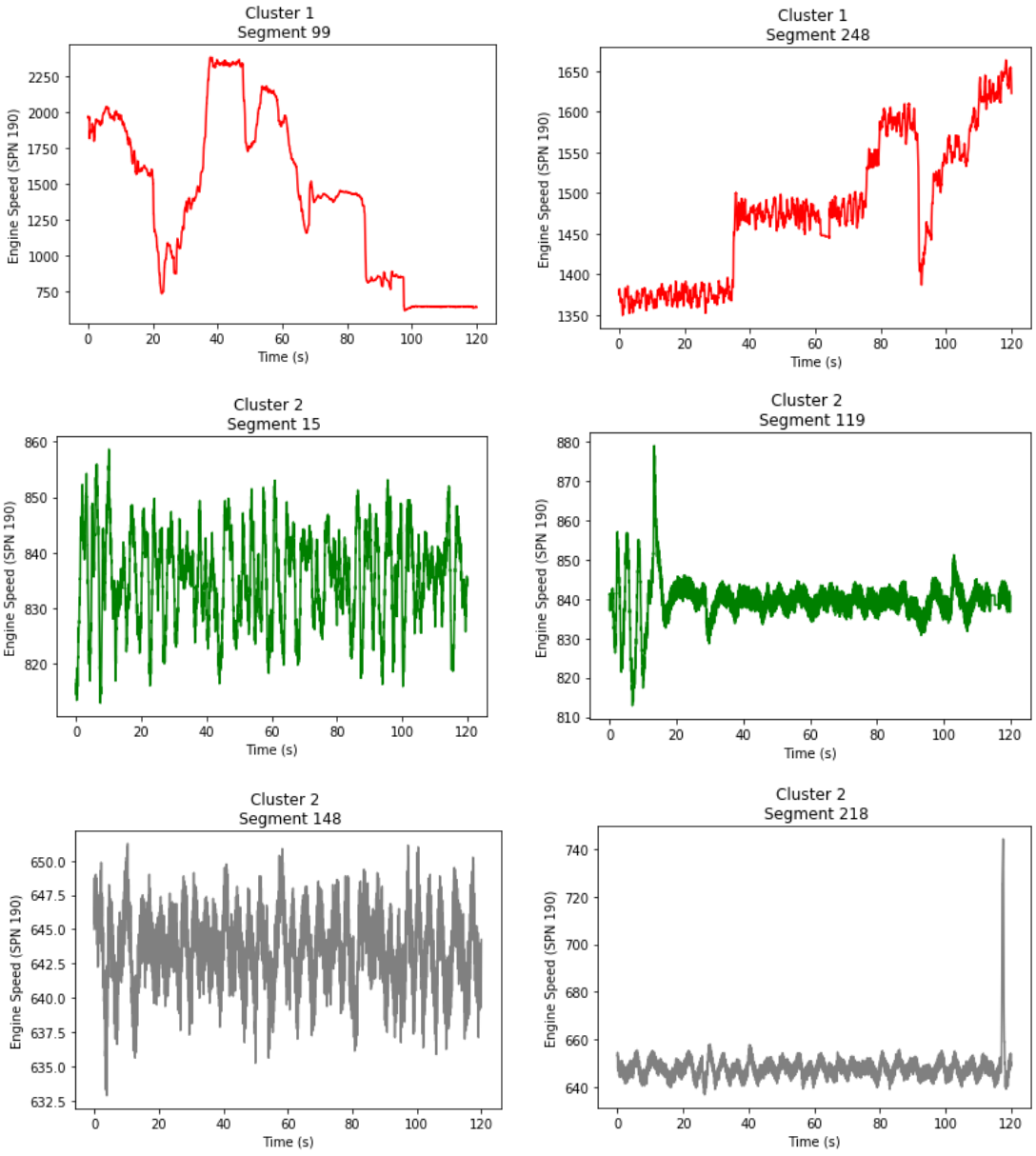

Figure 5.9: Representative segments for each cluster (Engine speed, SPN 190) Red segments, corresponding to cluster 2, represent the vehicle in overload. On the other hand, segments in green and gray belong to cluster 1, corresponding to a low workload. However, segments in green represents the vehicle while it is moving, while the ones in gray describe idle states.

workloads was 722.2 rpm, if a segment is characterized by a lower mean value, it is probably describing an idle state.

Therefore, combining the two cluster procedures it is possible to obtain three clusters, each characterized by a number of segments summarized in Table 5.8 and graphically represented in Figure 5.10: cluster 1 denotes working patterns or overload periods, cluster 2 describes moving patterns or regular workloads and cluster 3 contains segments associated with the vehicle in idle. As was expected, most of observations belong to cluster 3: indeed, as previously introduced, the vehicle under analysis is mainly used for testing devices and driver assistance systems, but rarely employed for working. Hence, it is plausible that the vehicle spent actually most of time in an idle state. On the other hand, observations in cluster 1 corresponds to a vehicle that is in a stressed state, identifying an overload usage. Finally, cluster 2 contains segments characterized by slow and not high variations, denoting a regular moving or working vehicle.

Cluster	Number of records
1	384
2	799
3	1222

Table 5.8: Number of records in each cluster obtained combining the two clustering strategies

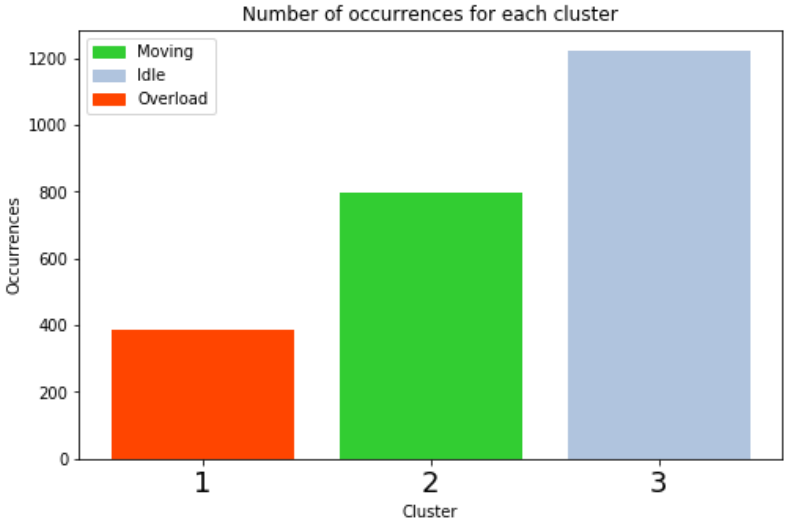

Figure 5.10: Number of segments in each cluster identified combining the two clustering strategies

The distributions of the features in the three different clusters can be visualized in Figure 5.11.

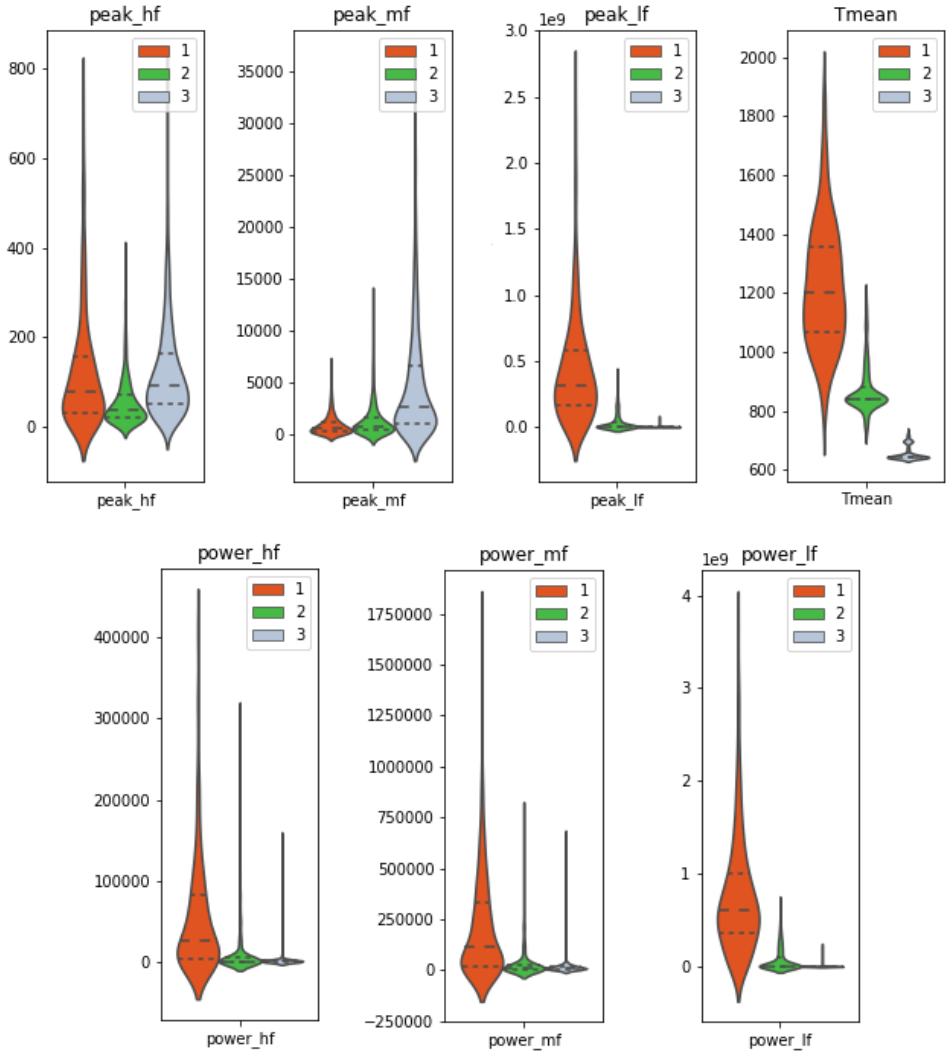

Figure 5.11: Violin plots showing the different distribution of frequency domain features between the 3 identified clusters

As it can be noticed, the distribution of features in cluster 3 is characterized by less variations with respect to the one associated with cluster 2, typically more centered on few values. In addition, it can be noticed how in some cases the distribution of observations in cluster 2 is more similar to the one associated with cluster 1, as for example in the case of the peak in medium frequencies, while other features, such as the power in the medium frequency, are characterized by

a distribution closer to the one associated with cluster 3. Cluster 2 represents, indeed, an intermediate situation between the observations assigned to *idle* and the ones associated with working patterns.

5.3.4 Result evaluation with NMEA 0183 messages data

As previously introduced, the problem under analysis is unsupervised. Hence, for validating the results obtained with the described procedure, some extra information are required: even if they have been already validated by domain experts, a numerical score can be obtained considering National Marine Electronics Association (NMEA) 0183 [3] messages data .

NMEA 0183 is a one-way serial data communication protocol, used for sending a subset of messages from the vehicle to external devices. This type of data contains extra information with respect to the ones provided by CAN bus data, such as the vehicle's position in terms of latitude and longitude and the vehicle's speed. These quantities, computed by the GPS receiver, can be used for evaluating results obtained in the previous sections. This type of information was not taken into account in the first part of this work since it was available only during the conclusion phase of this research. In addition, since GPS data are typically characterized by high measurement errors [37], they are considered too noisy to be included among input variables of a cluster algorithm. Even if they are only used for validation purposes, it should be kept in mind that the reliability of GPS data affects also the accuracy score obtained in the following. In addition, vehicle's speed can be used to validate idle state segments, but it is hard to make a distinction between moving and working states basing on it.
With these premises, cluster results are mapped into idle and not idle class.

During data import phase, since for each NMEA 0183 message a quality binary indicator is provided, messages characterized by quality indicator equal to zero are neglected, since this values indicates an invalid data. In addition, also the number of satellites in use is a parameter taken into account to select data with adequate quality.

In Figure 5.12 it is possible to see the behaviour of the parameter *speed over the ground* obtained from NMEA 0183 messages data. The collected speeds, ranging from 0 to 35 km/h, are coherent both with the type of vehicle under analysis, a heavy-duty vehicle expected to be characterized by quite slow movements, and the kind of performed tasks.

[3] www.nmea.org

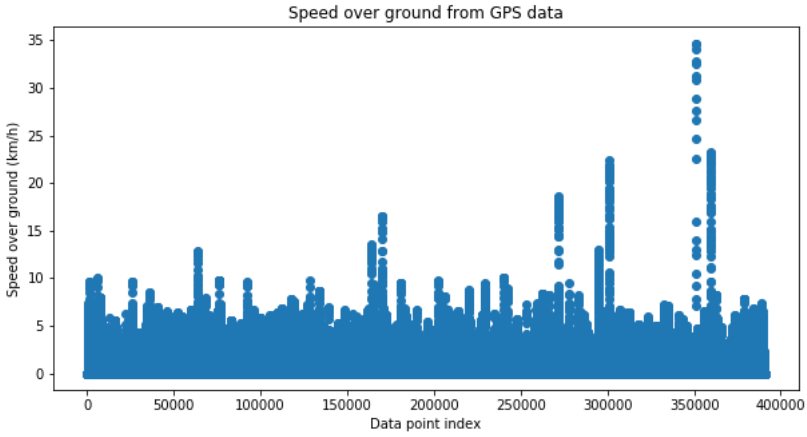

Figure 5.12: Speed over the ground (km/h) obtained from NMEA 0183 messages data

Computing the mean value of the vehicle's speed in each segment and combining this value with the number of non zero elements, it is possible to set a threshold to distinguish idle states from moving ones. The labels obtained with the described procedure are considered as ground truth values. The results can be evaluated inspecting the following confusion matrix:

		True labels	
		Idle	Moving/Working
Cluster outputs	Idle	1023	225
	Moving/Working	199	958

Table 5.9: Confusion matrix for evaluating the obtained results using as ground truth values the ones inferred using NMEA 0183 messages data

The accuracy score, defined as the ratio of corrected classified observations and the total number of items to be clustered, reacheas a value equal to 82.37%. This quantity does not focus on a class, but computes a value summarizing the overall class prediction quality [38]. Accuracy can be considered as a good measure for empirically evaluating the perform of the described method since the resulting dataset is quite symmetric, consisting in 50.81% of segments assigned to idle class according to GPS information and 49.19% to moving/working class.

For the sake of completeness, for evaluating separately the method with respect to each class, also sensitivity and specificity are computed [38], for this application

respectively defined as

$$\text{sensitivity} = \frac{\text{Number of observations correctly assigned to Idle}}{\text{Total number of Idle observations}}$$

and

$$\text{specificity} = \frac{\text{Number of observations correctly assigned to Moving/Working}}{\text{Total number of Moving/Working observations}}$$

The obtained values are

$$\text{sensitivity} = 83.72\% \quad \text{and} \quad \text{specificity} = 80.98\%$$

Hence, it is possible to conclude that the method seems to perform quite well both considering the overall accuracy and the described measures separately for each class. As previously introduced, these measures just give an indication about the method performance, since it is not possible to make a distinction between moving and working basing on the vehicle's speed and because of the GPS data measurement errors. In addition, by inspecting the wrongly assigned segments, it is possible to see that in some cases there are rapid and multiple changes in the engine speed, reaching also high values typically associated with a moving vehicle, even if the vehicle's speed is equal to zero for each observation recorded in the considered segment. Such a behaviour can be due to GPS data reliability, but can be also explained considering that the vehicle under analysis is used for testing purpose. In this sense, speed is not a reliable parameter for evaluating results, since high engine speed activity but with no movement associated can be considered as an error. However, this scenario should rarely occur once the procedure is applied to an actually working vehicle.

Chapter 6

Autoencoder-based deep learning for time series data clustering

In the previous chapters, a clustering technique based on the identification of a set of possible features is presented. However, because of the higher complexity of time series clustering due to the high dimensionality of the problem, conventional data analysis based on all the known features should be not enough. For this reasons, in this section a more challenging problem is addressed, namely the detection of hidden features in the available set of observations applying deep learning techniques. Indeed, there could exists some hidden features, due to both exogenous or endogenous factors, that cannot be directly detected. Their discovering allows typically to build more accurate model and highlight some underlying phenomena, but deep learning algorithms are designed for supervised problems, while time series data are often unlabelled [32]. A possible strategy to overcome this issue is presented in [32]. It is based on two consecutive stages: the former consists in extracting some features from the original data to summarize the behaviour of each segment to cluster. These features are used as inputs for a classical clustering algorithm, applied to obtain observations' labels and transform the original unsupervised problem into a supervised one. Then, in the second stage, it is possible to build an autoencoder-based deep learning model to predict the labels of previously unseen data, taking into account the information given both from known and hidden features. Following the describing procedure, the main issues related to time series clustering are solved: in the first step indeed the unlabelled data problem is overcome. In the last stage instead, neural networks are employed to simultaneously reduce the problem dimensionality and discover hidden features. The main drawback of this approach is that, since the classification neural network

is trained on the previously obtained class labels, the procedure is deeply dependent on the accuracy of unsupervised clustering results. Hence, it should be thought as a tool for validating cluster outputs or to highlight the presence of hidden features that should be included in the model, for example by inspecting items assigned to different clusters by the two procedures, rather than a tool for classification.

The idea of this chapter is then to apply the second stage described in [32] directly to the results obtained by clustering in the frequency domain: indeed, thanks to quite satisfactory results of previous sections both according to domain experts and with respect to GPS information, it could be a useful strategy to discover interesting characteristics that have not been taken into account before.

6.1 Artificial Neural Networks and Autoencoders

The proposed procedure is based on Autoencoders, a particular type of artificial neural networks (ANN).
Artificial Neural Networks are nowadays considered as one of the most advanced tools in data science [39] and are inspired by the learning procedure in the human brain. The structure of an artificial neural network is basically organized in different layers, namely an input layer, one or more hidden layers and an output layer. Each layer is characterized by nodes, also called *neurons*, used to represent the number of features. The number of nodes in a given layer defines its size. Furthermore, each node in the internal layers is equipped with an activation function (such as sigmoid, softmax or tangent hyperbolic). Then, information in each active neuron is mapped through links, also called *synapses*, from the input layer first to the nodes in the hidden layers and then to the output layer, which provides the result. The strength and the significance of the connections of each nodes is given by the weight associated to each link. They are used to identify the most important features to be considered. However, their values are typically unknown but iteratively learned and updated through a process called *law of learning* [39], basically consisting in solving an optimization problem. Indeed, during the network training, at every iteration weights are repeatedly assigned and adjusted in order to find the configuration that leads to the minimum training error. To find their optimal value, at each iteration, errors are back propagated into the network from the output layer towards the input one until the cost function is minimized and the model is then trained. This method, called backpropagation, requires labeled data to compare the output produced by the network with a ground truth value and back propagating the error: hence, it is used in supervised context.

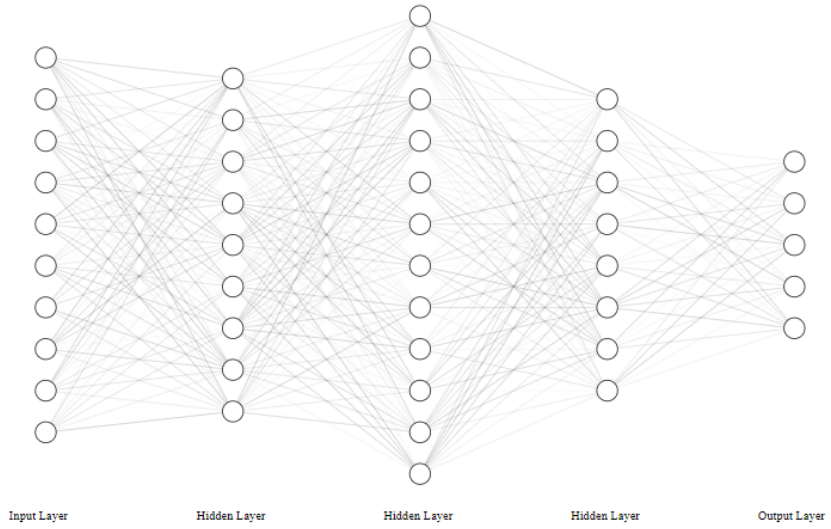

Figure 6.1: Artificial Neural Network architecture diagrams
The artificial neural network shown in Figure 6.1 is composed by an input layer consisting of 10 neurons, 3 hidden layers with respectively 9, 12 and 8 nodes and an output layer characterized by 5 nodes.
The figure is obtained using the online tool NN-SVG.

However, autoencoders are a type of neural networks that can be used in an unsupervised context since their output coincides with their input: the aim of autoencoders is indeed to reproduce their input.
An autoencoder is composed by two parts: an *encoder*, that maps the input data x into a lower dimensional set by identifying the most important features to describe the original set, and a *decoder* that has the role to reconstruct the original input from its reduced representation.
The output of the encoder is called *latent-space representation* and is an abstraction of the input data, a compress representation that allows simultaneously to summarize data and discover essential information that could be not inspectable in the original form [40]. Furthermore, decomposing the original data and then recovering it by means of the couple encoder-decoder is a commonly used procedure to remove surrounding noise [40].

From a mathematical point of view, an autoencoder network is a composition of two non-linear functions: an encoder function f that takes as input a data point $x \in \mathbf{R}^m$ and produces, as output, its latent-space representation $h \in \mathbf{R}^n$, where $n < m$, and a decoder function g that takes as input h and returns an output r. The objective of the network is to minimize the difference between the input and r.

More precisely, the relationship between input data and its latent-space representation can be written as [40]

$$h = f^k(x) = \sigma_0^k \left(W_0^k (f^{k-1}(x)) + b_0^k \right) \tag{6.1}$$

while the output can be written from its latent-space representation as

$$r = g^k(h) = \sigma_1^k \left(W_1^k (f^{k-1}(h)) + b_1^k \right) \tag{6.2}$$

where

- $k > 1$ is the number of hidden layers
- σ_0 and σ_1 represent activation functions. Commonly used choices are hyperbolic tangent, sigmoid or softmax function, depending on applications.
- W_0, b_0, W_1 and b_1 form the set of parameters that the model need to learn through back propagation algorithm, respectevely for the encoder and decoder. If W_0^l denotes the set of weights describing the strength of the links connecting layers $l-1$ and l, with $l \leq k$, then $W_0^l \in \mathbf{R}^{d_l \times d_{l-1}}$, where d_l is the dimension of data in layer l. In this case, since W_0 is one of the parameters used to characterize the encoder, data dimension is reduced from d_{l-1} to $d_l < d_{l-1}$. With the same notations, $b_0 \in \mathbf{R}^{d_{l-1}}$. Since autoencoder networks are assumed to be symmetric, the decoder will work analogously to the encoder but in the inverse order. Hence, $W_1^l \in \mathbf{R}^{d_{l-1} \times d_l}$ and $b_1 \in \mathbf{R}^{d_l}$. Furthermore, the number of layers and nodes of the decoder side will be the same of the encoding one [32].

W_0, b_0, W_1 and b_1 are chosen in order to find the best possible configuration, that is the one that optimizes a given loss function.
The aim of this type of analysis is to produce a latent-space representation of the input data such that the more important features of the given dataset can be extracted and used for further analysis.

There exists several types of auoencoders:

1. Basic autoencoder, characterized by an input layer with size $|x|$, a hidden layer with size $|h| < |x|$ and an ouput layer with size $|r| = |x|$.
2. Multilayer autoencoder, characterized by a number of hidden layers greater than 1. They are used when additional internal hidden layers are required to extract hidden features.
3. Convolutional autoencoder, in which the input is filtered for extracting only some parts of it. They are mainly employed in image processing applications.
4. Regularized autoencoder for which some other factors are considered in feature extraction and training, such as loss functions.

6.2 Encode-decoder-based deep learning method for time series clustering

As previously introduced, the method proposed in [32] is based on two phases, namely label generation and encode-decoder-based clustering.

The steps for label generation are summarized in the following. Given a set of time series segments,

1. The main features from the available set of data are extracted and used for building the *features vectors*, denoted as $f_1, f_2, \ldots, f_n$, for each segment.

2. A clustering algorithm to be applied to the extracted features is selected. For example, in [32], the conventional k-Means is used in this application. The outcome of this step is to identify the different cluster groups.

3. The obtained cluster labels are used to assign each segment to a class, transforming the given unsupervised problem into a supervised one.

This stage, summarized in Figure 6.2, has already been carried out in previous chapters.

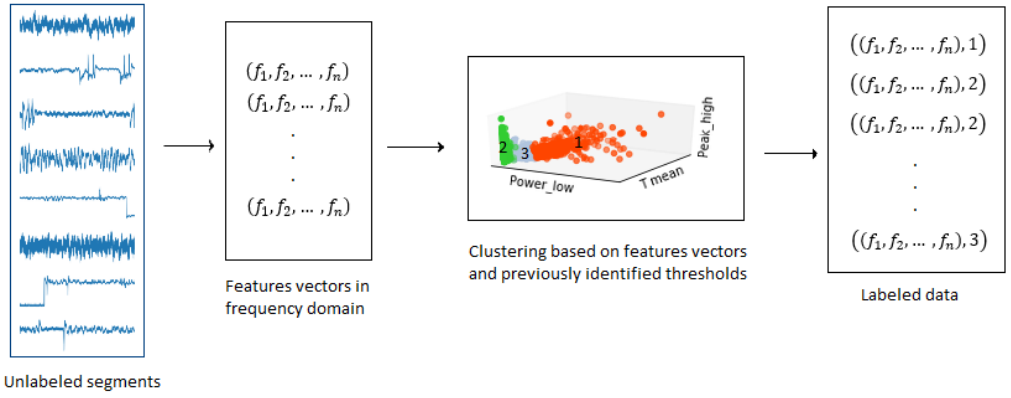

Figure 6.2: Labels generation main steps
The figure shows the main steps of labels generation, proposed in [32], referring to previous chapters results. More in details, SPN 190 segments and their relative previously obtained clusters are displayed.

Once the original unsupervised problem is mapped into a supervised one, encode-decoder-based clustering, whose steps are described in the following and graphically summarized in Figure 6.3 and 6.4, can be performed [32]. However, with respect to the procedure proposed in [32], some modifications have been made. More in

details, in this work, the autoencoder network training is performed separately and independently from the class labels prediction: instead of building a neural network that takes as input the features vector and the class labels and computes, as output, a single floating value used then to predict the class label, in this application an autoencoder is trained to transform the features vector into its latent-space representation. Then, the latent-space representation is used as input of a classification layer that produces as output a 3 dimensional vector, representing a posterior probability distribution over classes labels. Finally, each segment is assigned to the class for which the posterior probability is maximized.

With this procedure it is possible to train the autoencoder minimizing the Mean Square Error (MSE), while the classification layer is trained using as loss function the Categorical Cross Entropy (CCE), defined as

$$CCE(q, p) = -\sum_{x} p(x) \log(q(x))$$

where p denotes the ground truth distribution, while $q(x)$ is the predicted one.

These modifications allow to correctly treat the class label as a categorical attribute, instead of predicting a floating value mapped then into a categorical attribute as proposed in [32]. In this way it is possible to ignore the class labels during autoencoder training, avoiding bias results, and, most important, it prevents to make any mistakes introducing a not existing order relationship among classes [41] due to the mapping to integers values.

Summarizing, the step followed for encoder-decoder based clustering are

1. Split data into training, validation and testing set.
   In this application, the data split is performed in a stratify manner with respect to the previously obtained class labels. As result, 1611 (66.66%) segments are used to train the model, 531 (22.22%) to validate and 263 (11.11%) to test it.

2. Build an autoencoder neural network, with encoder side symmetric with respect to the decoder one.
   The details of the used architecture are described in the following.

3. Train the autoencoder on the train set and validate its hyperparameters on the validation set.

4. Use the trained network to obtain the latent-space representation for extracting the most important feature to summarize the input data, taking into account also the hidden features.

5. The latent-space representation is the input of a classification layer, whose output corresponds to a posterior probability distribution over classes labels.

6. Assign each segment to the class for which the probability is maximized.

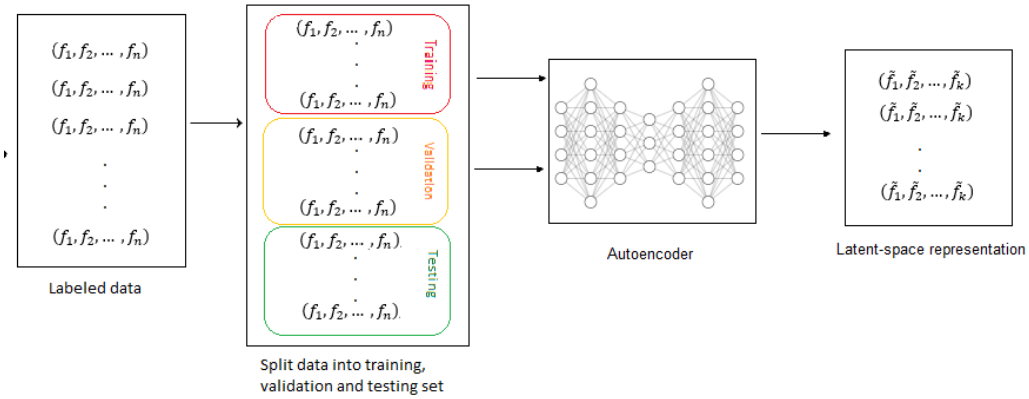

Figure 6.3: Main steps for extracting latent-space representation of input data

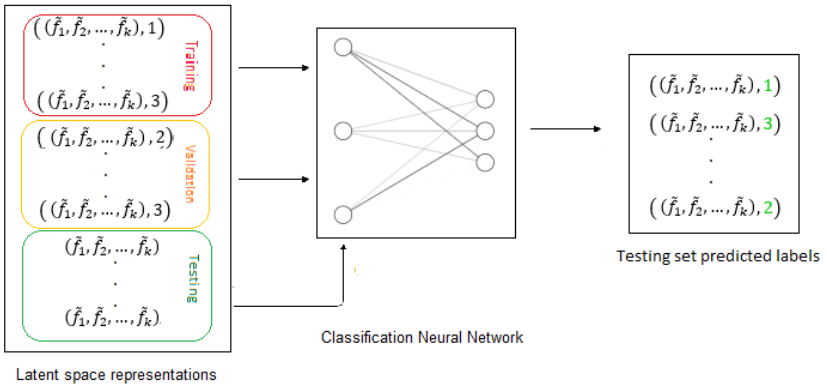

Figure 6.4: Main steps to classify segments based on the previous obtained latent-space representation

For what concerns the networks architecture, summarized in Figure 6.5, some of the specification given in [32] have been followed. More in details,

- The nodes characterizing the input layer represent the input features. Hence, in this application, the dimension of the input layer is 7.
- The encoder side is composed by 4 layers, with size respectively 100, 50, 20 and 3. The sizes decrease since one of the purpose for applying an encoding function is to reduce the dimensionality of input data and the size of the most hidden layer is equal to 3, the number of clusters identified in labels generation.
- The activation function used for the hidden layers is the Rectified Linear Unit ($ReLu$) defined as
$$ReLu(x) = \max(0, x)$$
From definition, it follows that ReLu activation function returns a value in the range $[0, +\infty)$ [42].
- By symmetry, the decoder side is characterized by the same structure of the encoder one, but in the inverse order.
- The output layer is composed by a number of nodes equal to the size of input data, 7 in this application.

Compiling the model, the closeness of the output to the input vector is measured using the mean square error (MSE).

To compile the model, the Adam optimized has been chosen. It is a stochastic first-order gradient based optimization algorithm, used when the objective function is characterized by a high number of parameters or for solving noisy problems [43]. Furthermore, it has gained popularity because it has been proved that it achieves good results in fewer iterations with respect to other commonly used stochastic optimization methods [43]. It is based on adaptive estimation of moments using bias-corrected moving averages, hence its name.

The chosen optimization algorithm works separately on batch of training data, randomly selected and used for training until it exhausted the entire available dataset, completing an epoch. Then, this procedure is repeated an user specified number of epochs [44]. The number of epochs and the batch size represent some of the model hyperparameters. The size of the batch should be large enough to correctly approximate the behaviour of the entire dataset [44], but it has been proven that bigger batch sizes lead to models that are hard to generalize [45]. Indeed, it is possible to see this behaviour inspecting the validation error associated with a batch size of 1024, as the one suggested in [32], compared with the validation

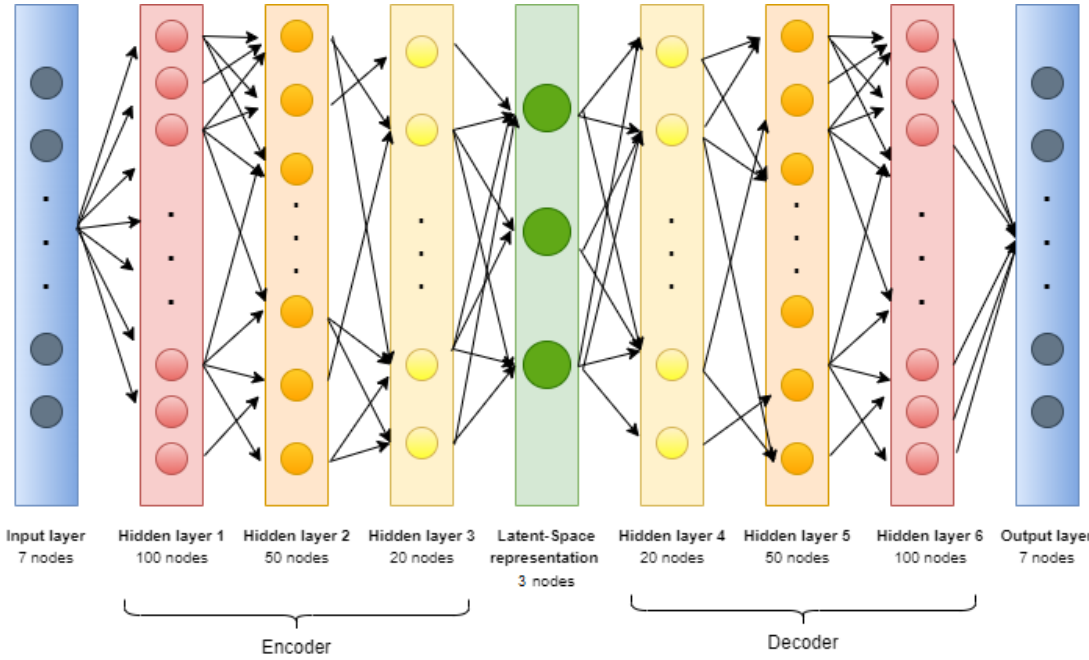

Figure 6.5: Encoder-Decoder based network architecture
The figure shows the architecture of the network trained for applying the encoder-decoder-based deep learning method for time series clustering..

error in the case of the batch size selected for this application, 256. Decreasing the batch size, the validation error get closer to the training error, meaning that the model is less overfitting the training data.

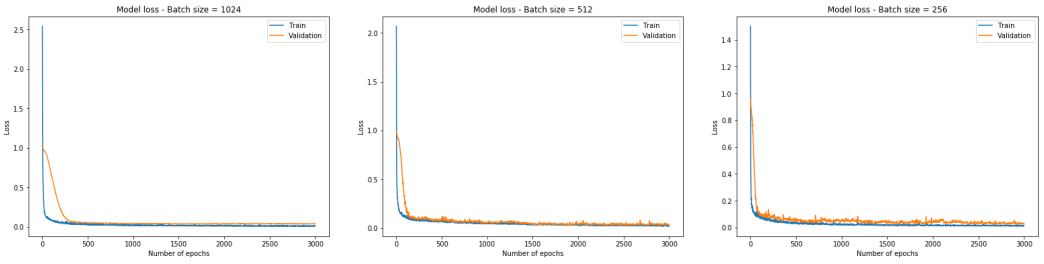

Figure 6.6: Loss function with respect to the number of epochs for different batch sizes, namely 1024, 512 and 256

From Figure 6.6, it is also possible to see the behaviour of MSE with respect to the number of epochs used to train the network. As in the previous case, the number of epochs should be chosen on the basis of the validation set. It represents the number of iterations used to update the weights [46]. The model has been

trained for 3000 epochs, but as it is possible to be noticed, after less than 500 epochs the MSE is close to 0 both for the training and the validation set. Hence, a lower value can be selected in order to prevent overfitting and reduce computational cost.

Once the latent-space representations have been obtained, they are used as input for a classification layer: it is a 3 nodes layer that produces, as output, a probability distribution over the 3 identified classes.
To this purpose, the activation function associated with this layer is the *Softmax*, widely used to obtain a categorical probabilistic distribution for classification from the output of a neural network [47], [48]. Given $x \in \mathbf{R}^n$, a softmax function $\sigma : \mathbf{R}^n \longrightarrow \mathbf{R}^n$ is defined as

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \qquad \text{for i} = 1, \ldots, \text{n}.$$

As it is possible to be noticed from definition, softmax function takes as input the components of vector x and maps them into values in the range (0,1) such as they sum up to 1. The output of softmax function can be then interpret as a posterior probability: the i-th component represents the probability that the class label y is i, given x [48], namely

$$\sigma(x)_i = p(y = i|x)$$

Finally, the predicted class label is the one that maximizes the obtained posterior probability. This final step is justified by the empirically proved connection between standard K-Means algorithm and classification performed with a classification layer equipped with softmax activation function [49], whose mathematical proof is achieved in [50].

Applying the described procedure, the 96,96% of testing observations are assigned to the same cluster obtained with the previous clustering method. However, as it is possible to be noticed in Figure 6.7, 8 of them are assigned to a different class. These segments may indicate that the autoencoder-based cluster is performed taking into account some hidden features that are not considered by the k-Means algorithm [32].

Some of the segments assigned to different classes by the two method are shown in Figure 6.8.

It can be seen in Figure 6.8 that segments 555 and 1666, both previously identified as moving or regular working segments, are now assigned to cluster 1, composed by higher working states. Indeed, in both cases, it is possible to see a short period characterized by variations that are more likely to be associated with higher working states rather than regular working or moving patterns. However,

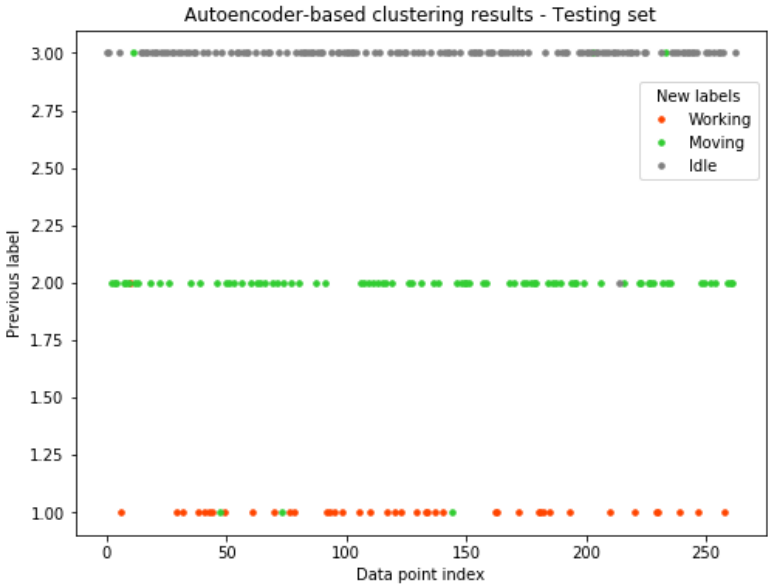

Figure 6.7: Class labels obtained applying the autoencoder-based method to the testing set, compared with the previously obtained ones

The picture shows the segments assigned to different classes in the two described procedure: indeed, the y axis refers to the class labels obtained with the previous clustering method (1 for working, 2 for moving/regular working and 3 for idle), while the points color highlight the class labels obtained with the autoencoder-based clustering

before and after this short period, the engine speed shows a regular behaviour. They represent hence a kind of intermediate situation and some hidden features made them to be assigned to cluster 1 instead of cluster 2.

Conversely, segment 2247, previously assigned to cluster 1, is now associated with cluster 2: in this case, even if it is possible to notice some variations typically associated with higher working states, the engine speed is in most of observations quite regular, hence assigned to cluster 1.

Finally, segment 1995 was previously described as an idle segment, while the autoencoder-based cluster results assign it to cluster 2. According to domain experts, the behaviour of engine speed shown for segment 1995 is typically associated with a vehicle that, from an idle state, starts to move. Hence, even if for most of observation the vehicle is in an idle state, it is not wrong to group this segments with the ones associated with moving state.

These qualitatively described differences can be quantitatively inspected looking at the values taken by the features for segments 555, 1666, 2247 and 1995 and comparing them with the ranges of variability of these quantities in their respective previously assigned clusters.

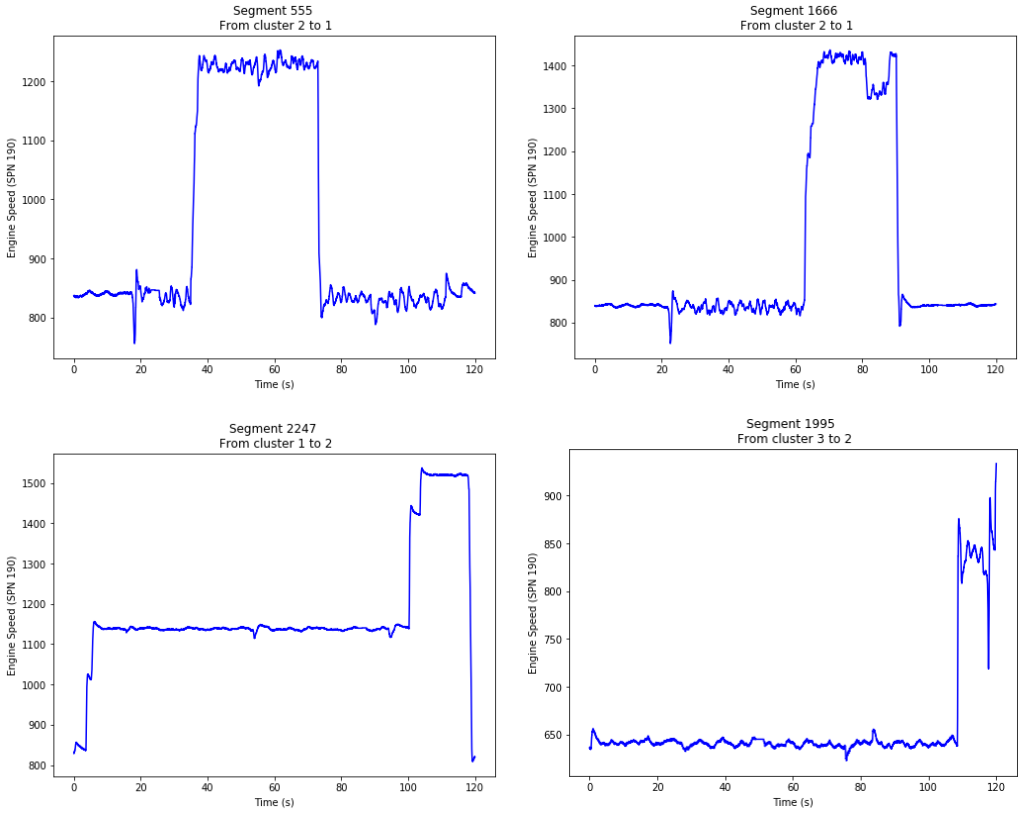

Figure 6.8: Some of the segments (namely 555, 1666, 2247 and 1995) assigned to different classes by the autoencoder-based cluster method with respect to the previous proposed one.

Indeed, as can be noticed from Figure 6.9, segments 555 and 1666 are characterized by a value of the peak in the low frequencies closer to the ones taken by items belonging to cluster 1, displayed in orange, rather than the ones taken by cluster 2, displayed in green.
The same conclusions hold also for the power in the low frequencies, where it is possible to note that segment 1666 takes a value strictly greater than the values taken by segments in cluster 2.

Similarly, segment 2247 is characterized by peak value in the low frequencies and power in the middle and high frequencies closer to the values taken by the same feature for segments associated with the moving state.

Finally, the power in the middle and high frequencies associated with segment 1995 is strictly greater than the maximum value taken by this quantity for items belonging to cluster 2 and closer to the values associated with cluster 1.

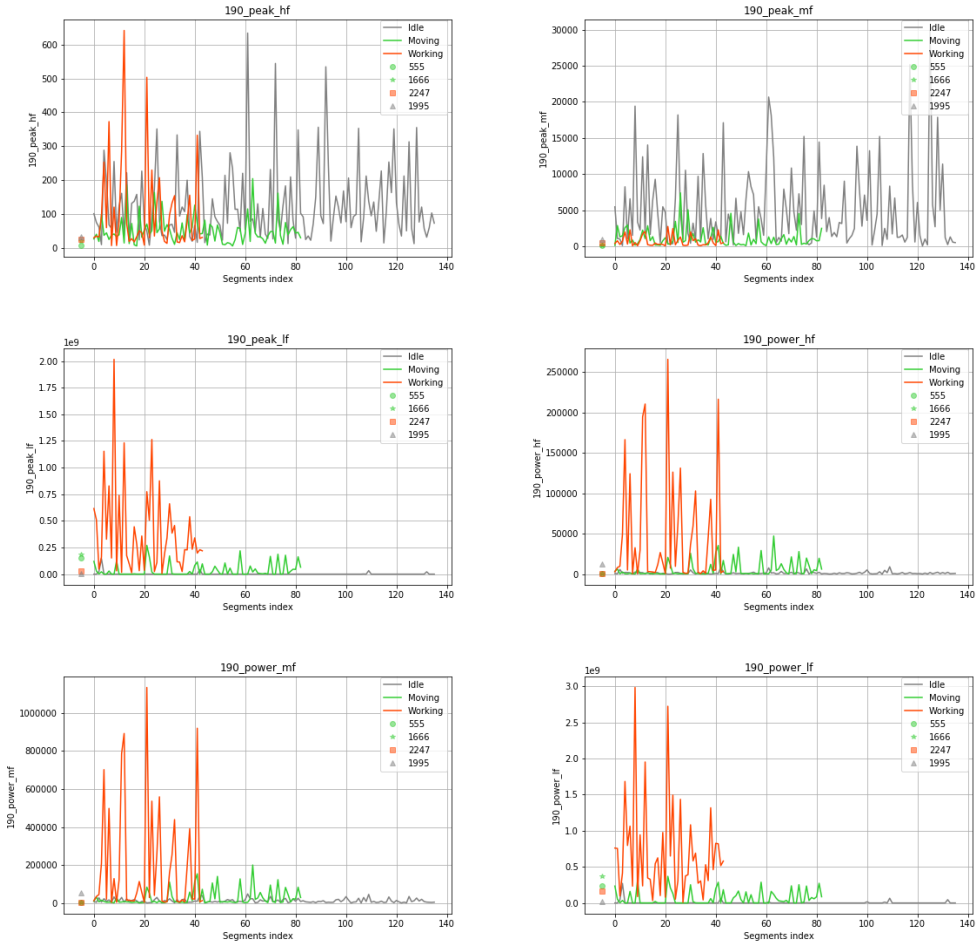

Figure 6.9: Comparison between the features ranges of variability in the different clusters and the values taken by these quantities in the case of differently assigned segments.
The colors assigned to the differently assigned segments refer to the classes obtained by the first procedure, before the correction made applying the autoencoder-based procedure.

In conclusion, it is possible to say that the identified segments can be considered, in some sense, as outliers or extreme observations in the cluster where they were previously assigned. Their behaviour can be explained looking at the values they take in the time domain. Indeed, in all the presented cases, the engine speed shows a behaviour that does not identify a single state, but contains elements both of the previous identified class, both of the new one. This type of analysis suggests

hence that some other features should be considered for correctly identifying this segments too. For example, the peaks, the powers and the temporal mean could be computed not overall the 2 minutes points, but considering shorter intervals in the same segment. For the same reasons, the procedure may report that a fixed window segmentation could be not appropriate for these cases: hence, other techniques based on variable length time series windowing, could be applied in order to identify homogeneous segments in the original time series.

Chapter 7

Conclusions and future works

Being able of automatically identifying usage patterns and workload states for heavy duty vehicles is extremely important for companies which desire to monitor their equipment. Analysing CAN bus data represents a way to provide support for optimizing maintenance, production, business and investments. Furthermore, deeply studying the properties of the signals generated at high frequencies by sensors installed on the vehicle, it is possible to identify the most relevant signals and the most appropriate sampling rate to use.

In the first part of the thesis work, a procedure based on Fourier transform for resampling a CAN bus signal to a given constant rate is presented: it is used, in this real case application, for synchronizing CAN bus data, transmitted at different and irregular rates because of the inherent structure of CAN networks. Having synchronized and evenly spaced data points is a requirement for many algorithms, as the ones used in the following of the thesis work, since it allows to treat SPNs measurements as components of a multivariate time series.
The described procedure is general and can be used both for upsampling or downsampling. In addition, it could be a tool for changing data granularity, since working with ultrafine data could be redundant and noisy for some SPNs or applications. Indeed, as highlighted in the first part of the thesis work, some SPNs are characterized by constant measurements all over the available time windows, while others are characterized by slow changes. In this cases, lower sampling rates should be set in order to reduce the transmission costs. In any case, constant SPNs should be identified and reported because they could represent technical failures (due to sensors, device or transmission) that should be checked by the device's manufacturer.

The resampling procedure is not suitable for too irregular sampled SPNs: however, since it is not clear to domain experts if this behaviour is due to technical errors or because irregular sampled parameters are malfunction tell-tales, they are not considered until further information are provided.

Once signal synchronization is achieved, SPNs can be deeply studied in order to describe their properties both treating each parameter individually, since they are describing different physical quantities, both considering their mutual interaction, since their are describing the behaviour of the same vehicle. As result, the most important parameters to be monitored are identified, removing SPNs that are not informative, correlated and with particular internal structures, such as the ones characterized by strong (and trivial) positive trend.
Reducing the number of variables to collect, send on the network and store, it is possible to limit data transmission costs. For the results obtained in this thesis, the number of monitored parameters can be reduced from 20 to 4, namely the engine coolant temperature, the engine speed, the engine fuel delivery pressure and the charge pressure.

Finally, two unsupervised clustering procedures are described in order to automatically detect usage patterns in data. The first one exploits the data synchronization to $1Hz$, considering each line of the dataset as an item to be clusterized. Even if this approach does not take into account the temporal relationship between observations, its outcome is to automatically identify thresholds to divide different workload types basing on the instantaneous values taken by SPNs. As result, two different workloads are identified, namely overload and idle or standard workload. The two detected states seem to be coherent with the task carried out by the vehicle, but since no ground truth activity is provided, there is no measure to quantitatively validate results. It has been asked to the testing field to collect few days of observations providing a report describing the tasks performed by the vehicle. In this case, it should be possible to use this limited amount of observations as validation set.

The resulting thresholds identify states that are analogous to the manually set ones used by the company. However, it is not possible to make a comparison with the actually employed thresholds since they are based also on digital input data that are not available for this analysis. As future improvement, an additional device could be installed on the vehicle to include also digital inputs among clustering variables. Furthermore, by varying the tasks carried out by the vehicle, a higher number of identified states are expected.

The second method for identifying usage patterns proposed in this thesis is based on signals spectral content. Indeed, analysing the available measurements in the frequency domain it is possible to describe the signals' variations by means of their frequencies distribution. Since this type of analysis is suitable only for signals

with enough informative spectral content, spectral analysis is performed to identify signal characterized by high variations and enough resolution. From its outcome it is possible to obtain some guidelines for SPNs sampling rates, that could be lowered without loosing much information for some SPNs, are adequate for some signals while the resolution should be higher for others.

The SPNs spectral content, separately for low, medium and high frequencies, is then used to summarize the behaviour of the vehicle every 2 minutes of observations. The time window size is set combining the range provided by domain experts with an application of VALMOD algorithm.
This method does not require time series synchronization and takes into account the temporal order among observations. Its outcome is to automatically tell a highly variations segment, describing a higher workload or a more aggressive driving style, with multiple breaks and acceleration, from a slow varying and more regular one, describing an idle state or a regular driving style. Its result consists in dividing segments according to the main state that characterizes and summarizes the behaviour of the vehicle in the 2 considered minutes. Once again, the obtained clusters seem to be coherent with the tasks carried out by the vehicle, but no quantitative measure can be used to validate results since it is a total unsupervised context. However, what can be immediately noticed is that, since regularly moving segments are characterized by shapes similar to idle ones, this method is not able to tell when the vehicle is moving. The difference among the two states is the engine speed value: in the first case, it would be higher, while if the vehicle is idle, it would be around the minimum value. Applying the thresholds identified by the previous method, it is possible to obtain three different clusters, describing idle state, moving/regular working and higher workload.

The obtained results have been validated by domain experts and are quite coherent with the ones expected, considering the particular tasks the vehicle carried on during data collection. However, the described techniques are general and they are expected to perform well also on more realistic data, with a higher number of SPNs used as variables and with data coming from several vehicles.
Combining results with extra information obtained from GPS data, it is possible to conclude that this procedure performs pretty well, being able of telling an idle state from a moving/working one reaching an estimated accuracy score over 82%. This value is quite high considering the limited reliability of GPS data and the vehicle's irregular behaviour due to the particular tasks it carried out during data collection phase.

To further validate the obtained results, an autoencoder-based deep learning technique for time series data clustering has been applied: inspecting segments that are assigned to a different classes by the autoencoder-based clustering with respect to the ones obtained with the previous described procedure, it is possible to identify the presence of some hidden features, not previously taken into account by

the standard k-Means algorithm. For example, this procedure reveals the presence of not homogeneous segments that can be assigned to different classes because they represent intermediate situations, such as a vehicle that from idle state starts to move. Hence, it suggests possible modification to improve clustering results: in the first place, the same features could be computed for a given segment separately in shorter time intervals, such as $30 - 45$ seconds. Alternatively, another approach for time series segmentation could be applied: indeed, these results could report that a fixed window segmentation is not appropriate because in this application it leads to not homogeneous segments. These problems could be fixed, for example, applying variable length techniques.

Even if available data were collected during a limited number of performed tasks, significant results have been achieved and all the described methods and procedures can be easily generalized and applied to a larger set of monitored parameters, collected in standard working situations.

This thesis represents hence just a starting point and several improvements can be made. At first, the optimal data granularity can be searched, both tuning it accordingly to spectral analysis results, both evaluating differences by means of a loss function lowering the sampling rate. In addition, also several aggregation techniques could be considered as well as resampling using different bases, such as cosine or wavelet. Using different basis decompositions, it is also possible to identify different features to summarize the behaviour of SPNs in a fixed length window. Finally, Recurrent Neural Network or Long Short-Term Memory Network can be applied to the original segments to perform time series clustering without the need of computing features and exploiting hidden features and the potentials of deep learning [7].

Bibliography

[1] Shai Ben-David and Shai Shalev-Shwartz. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014 (cit. on pp. 1, 59, 61, 62).

[2] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020 (cit. on pp. 1, 2).

[3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN: 026201825X (cit. on p. 1).

[4] Dena Markudova, Elena Baralis, Luca Cagliero, Marco Mellia, Luca Vassio, Elvio Amparore, Riccardo Loti, and Lucia Salvatori. «Heterogeneous Industrial Vehicle Usage Predictions - A Real Case». In: *EDBT/ICDT Workshops* (2019) (cit. on pp. 3, 8, 78).

[5] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. «Using Data Analytics to Detect Anomalous States in Vehicles». In: *CoRR* abs/1512.08048 (2015) (cit. on p. 5).

[6] Peng Zhang. *Advanced industrial control technology*. William Andrew, 2010 (cit. on p. 5).

[7] Thomas Huybrechts, Yon Vanommeslaeghe, Dries Blontrock, Gregory Van Barel, and Peter Hellinckx. «Automatic Reverse Engineering of CAN Bus Data Using Machine Learning Techniques». In: (Jan. 2018), pp. 751–761 (cit. on pp. 6, 18, 34, 69, 105).

[8] Umberto Fugiglando, Emanuele Massaro, Paolo Santi, Sebastiano Milardo, Kacem Abida, Rainer Stahlmann, Florian Netter, and Carlo Ratti. «Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment». In: *CoRR* abs/1710.04133 (2017) (cit. on pp. 11, 18).

[9] Rabia Korifi, Yveline Le Dréau, and Nathalie Dupuy. «Comparative study of the alignment method on experimental and simulated chromatographic data». In: *Journal of separation science* 37 (Nov. 2014). DOI: 10.1002/jssc.201400700 (cit. on p. 18).

[10] Martin Vetterli, Jelena Kovacevic, and Vivek K. Goyal. *Foundations of Signal Processing*. Cambridge Univ. Press, 2014 (cit. on pp. 19–21, 23, 25–28, 33, 34).

[11] Alessandro Falaschi. *Trasmissione dei Segnali e Sistemi di Telecomunicazione*. 2019 (cit. on pp. 20, 25, 75, 77).

[12] S. G. Mallat. «A theory for multiresolution signal decomposition: the wavelet representation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 674–693 (cit. on p. 22).

[13] Edward Bullmore, C Long, John Suckling, J Fadili, Gemma Calvert, F Zelaya, Thomas Carpenter, and M Brammer. «Colored noise and computational inference in neurophysiological (fMRI) time series analysis: Resampling methods in time and wavelet domains». In: *Human brain mapping* 12 (Feb. 2001), pp. 61–78 (cit. on p. 22).

[14] Gilbert Strang. «Wavelet transforms versus Fourier transforms». In: *Bulletin of the American Mathematical Society* 28.2 (Apr. 1993), pp. 288–306. ISSN: 0273-0979. DOI: 10.1090/s0273-0979-1993-00390-2 (cit. on p. 23).

[15] Katarzyn J. Blinowska and Jaroslaw Zygierewicz. *Practical Biomedical Signal Analysis Using MATLAB*. 1st. USA: CRC Press, Inc., 2011. ISBN: 1439812020 (cit. on pp. 34, 53, 54, 56).

[16] Richard G. Lyons. *Understanding Digital Signal Processing*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1996. ISBN: 0201634678 (cit. on p. 35).

[17] J.O. Smith. *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications*. W3K Publishing. BookSurge Publishing, 2007. ISBN: 9780974560748 (cit. on pp. 35, 36).

[18] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. USA: California Technical Publishing, 1997. ISBN: 0966017633 (cit. on p. 36).

[19] A. Pal and PKS Prakash. *Practical Time Series Analysis*. Packt Publishing, 2017 (cit. on pp. 38, 41–43).

[20] P.J. Brockwell and R.A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, 2016. ISBN: 9783319298542 (cit. on pp. 38–40).

[21] Gebhard Kirchgässner and Jürgen Wolters. *Introduction to Modern Time Series Analysis*. Springer-Verlag Berlin Heidelberg, 2007 (cit. on pp. 39–41).

[22] Mike West. «Time series decomposition». In: *Biometrika* 84.2 (June 1997), pp. 489–494. ISSN: 0006-3444. DOI: 10.1093/biomet/84.2.489 (cit. on p. 42).

[23] Janos Abonyi and Balazs Feil. *Cluster Analysis for Data Mining and System Identification*. Birkhäuser Basel, 2007. ISBN: 3764379871 (cit. on pp. 43, 70).

[24] Timothy R Derrick and Joshua M. Thomas. «Time Series Analysis: The Cross-Correlation Function». In: Kinesiology Publications, 2004 (cit. on p. 49).

[25] Raquel Prado and Mike West. *Time Series: Modeling, Computation, and Inference*. 1st. Chapman Hall/CRC, 2010. ISBN: 1420093363 (cit. on p. 49).

[26] C. Marzban, Paul R. Illian, David Morison, and P. D. Mourad. «Within-group and between-group correlation : Illustration on noninvasive estimation of intracranial pressure». In: (cit. on pp. 50, 51).

[27] Sheldon M Ross. *Introductory Statistics*. Academic Press, 2010 (cit. on p. 58).

[28] Karthik Ramasubramanian and Abhishek Singh. *Machine Learning Using R: With Time Series and Industry-Based Use Cases in R*. Jan. 2019. ISBN: 978-1-4842-4214-8. DOI: 10.1007/978-1-4842-4215-5 (cit. on pp. 59, 60).

[29] Preeti Arora and Shipra Varshney. «Analysis of K-Means and K-Medoids Algorithm For Big Data». In: *Procedia Comput. Sci.* 78.C (Mar. 2016), pp. 507–512. ISSN: 1877-0509. DOI: 10.1016/j.procs.2016.02.095 (cit. on p. 60).

[30] Rajan Chattamvelli. *Data Mining Algorithms*. 1st. Alpha Science International, Ltd, 2011. ISBN: 1842656848 (cit. on pp. 60, 61).

[31] Paweł Cichosz. «Hierarchical clustering». eng. In: *Data Mining Algorithms*. Chichester, UK: John Wiley Sons, Ltd, 2015, pp. 349–372. ISBN: 9781118332580 (cit. on p. 60).

[32] Neda Tavakoli, Sima Siami Namini, Mahdi Khanghah, Fahimeh Soltani, and Akbar Siami Namin. «An autoencoder-based deep learning approach for clustering time series data». In: *SN Applied Sciences* 2 (May 2020). DOI: 10.1007/s42452-020-2584-8 (cit. on pp. 69, 88, 89, 91–93, 95, 97).

[33] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh. «VALMOD: A Suite for Easy and Exact Detection of Variable Length Motifs in Data Series». In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. Houston, TX, USA: Association for Computing Machinery, 2018, pp. 1757–1760. ISBN: 9781450347037. DOI: 10.1145/3183713.3193556. URL: https://doi.org/10.1145/3183713.3193556 (cit. on pp. 69–72).

[34] Janos Abonyi, Balazs Feil, S. Katalin Nemeth, and Peter Arva. «Principal Component Analysis based Time Series Segmentation - A New Sensor Fusion Algorithm». In: 2004 (cit. on p. 70).

[35] A. Galka. *Topics in Nonlinear Time Series Analysis: With Implications for EEG Analysis*. Advanced series in nonlinear dynamics. World Scientific, 2000. ISBN: 9789810241483 (cit. on p. 75).

[36] Bobo Zhao, Zhu Wang, Zhiwen Yu, and Bin Guo. «EmotionSense: Emotion Recognition Based on Wearable Wristband». In: Oct. 2018, pp. 346–355. DOI: 10.1109/SmartWorld.2018.00091 (cit. on p. 77).

[37] Jing Li, George Taylor, and David Kidner. «Accuracy and reliability of map-matched GPS coordinates: The dependence on terrain model resolution and interpolation algorithm». In: *Computers Geosciences* 31 (Mar. 2005), pp. 241–251. DOI: 10.1016/j.cageo.2004.06.011 (cit. on p. 85).

[38] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. «Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation». In: vol. Vol. 4304. Jan. 2006, pp. 1015–1021. DOI: 10.1007/11941439_114 (cit. on p. 86).

[39] Enzo Grossi and Massimo Buscema. «Introduction to artificial neural networks». In: *European journal of gastroenterology hepatology* 19 (Jan. 2008), pp. 1046–54 (cit. on p. 89).

[40] Min Chen, Xiaobo Shi, Yin Zhang, Di Wu, and Mohsen Guizani. «Deep Features Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network». eng. In: *IEEE Transactions on Big Data* PP.99 (2017), pp. 1–1. ISSN: IEEE Transactions on Big Data (cit. on pp. 90, 91).

[41] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. ISBN: 1461471370 (cit. on p. 93).

[42] Abien Fred Agarap. «Deep learning using rectified linear units (relu)». In: *arXiv preprint arXiv:1803.08375* (2018) (cit. on p. 95).

[43] Diederik P Kingma and J Adam Ba. «A method for stochastic optimization. arXiv 2014». In: *arXiv preprint arXiv:1412.6980* 434 (2019) (cit. on p. 95).

[44] Michael A Nielsen. *Neural networks and deep learning*. Vol. 2018. San Francisco: Determination press, 2015 (cit. on p. 95).

[45] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. «On large-batch training for deep learning: Generalization gap and sharp minima». In: *arXiv preprint arXiv:1609.04836* (2016) (cit. on p. 95).

[46] Kevin Gurney. *An Introduction to Neural Networks*. USA: Taylor Francis, Inc., 1997. ISBN: 1857286731 (cit. on p. 96).

[47] Zhenyue Qin and Dongwoo Kim. «Rethinking Softmax with Cross-Entropy: Neural Network Classifier as Mutual Information Estimator». In: *ArXiv* abs/1911.10688 (2019) (cit. on p. 97).

[48] Mona ElBedwehy, G.M. Behery, and Reda Elbarougy. «Face Recognition Based on Relative Gradient Magnitude Strength». In: *ARABIAN JOURNAL FOR SCIENCE AND ENGINEERING* (May 2020). DOI: 10.1007/s13369-020-04538-y (cit. on p. 97).

[49] Ozsel Kilinc and Ismail Uysal. «Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization». In: *arXiv* (2018) (cit. on p. 97).

[50] Sibylle Hess, Wouter Duivesteijn, and Decebal Mocanu. «Softmax-based Classification is k-means Clustering: Formal Proof, Consequences for Adversarial Attacks, and Improvement through Centroid Based Tailoring». In: *arXiv preprint arXiv:2001.01987* (2020) (cit. on p. 97).