Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (*XXXV*cycle)

Machine Learning with Limited Label Availability

Algorithms and Applications

By

Flavio Giobergia

******

Supervisor:
Prof. Elena Baralis

Doctoral Examination Committee:
Prof. Sara Comai, Referee, Politecnico di Milano, Italy
Dr. Dino Ienco, Referee, INRAE, France
Prof. Rosa Meo, Università degli studi di Torino, Italy
Dr. Genoveva Vargas-Solar, CNRS, France
Prof. Silvia Chiusano, Politecnico di Torino, Italy

Politecnico di Torino
2023

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Flavio Giobergia
2023

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

A few words in a page of a thesis is definitely not enough credit for all the people who helped me along this journey, but that's all I can spare for the time being, so hopefully that's enough. If not, the next round of beers will be on me.

The one person without whom this thesis (and this journey) would not have been possible is my supervisor, Prof. Elena Baralis, who believed in me from the beginning and who helped and guided me along the way. Thank you.

I would also like to thank and acknowledge the reviewers of this work, Prof. Sara Comai and Dr. Dino Ienco, for taking the time to help me achieve this accomplishment: your effort has been very much appreciated.

Next (not that order matters), I believe a debt of gratitude is due to my colleagues and friends at Lab 5. Each of you contributed in your own way to make this journey a little easier with each lunch, each coffee break, each whiteboard session and, of course, each aperitif. I will single out Eliana and Giuseppe, for having also been amazing pandemic-grade flatmates.

My family probably needs the biggest thanks of them all. My parents and my sister have been there for me literally my entire life. I cannot overstate how much that means to me, and for that I will always be grateful.

Thank you Monica for, you know, everything. Putting up with me isn't easy, but you make it look like a walk in the park. How you do that is beyond my understanding.

Finally, I'd like to thank all the friends I met along the way: be it in high school, university, random encounters, or anywhere else; you have all helped me become who I am today.

Abstract

An underlying assumption typically made about machine learning algorithms is that the data required for training is both available and curated. However, this assumption does not often hold outside of very specific ad-hoc settings. This in turn hinders the chances of deploying robust machine learning models to address real-world problems, thus reducing the impact that the entire field could actually have.

One of the main obstacles to the successful application of many machine learning algorithms stems from a lack of supervised data: if labels are unavailable, or only available in limited quantities, supervised learning algorithms can struggle during the process of learning useful patterns.

This work of thesis acknowledges the problems that come with limited label availability scenarios and offers contributions within the field. In particular, three main areas of interest are identified, where (i) labelled data is completely unavailable – in which case only unsupervised techniques can be applied, (ii) labelled data is only available in domains other than the one of interest, in which case useful information can still be learned in the label-rich domain and propagated to the label-scarce one and (iii) only reduced amounts of labelled data is available, along with large quantities of unlabelled data.

The thesis addresses these aspects by proposing contributions from both an algorithmic and an applied perspective. In particular, the following contributions that address the aforementioned areas are introduced: (i) improving the training process of existing unsupervised algorithms to reduce their computational complexity, (ii) developing a methodology for propagating representations learned in label-rich domain to label-scarce ones, by means of aligned latent spaces, and (iii) introducing a semi-supervised learning algorithm that learns from self-assigned pseudo-labels based on the introduction of an explicit confidence mechanism.

The proposed contributions are evaluated from both theoretical and empirical perspectives, discussed and contextualized within the existing state of the art. We additionally extensively cover an applied use case, where labels are originally not available but can be inferred in a semi-supervised way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the field of machine learning has gained significant momentum. Algorithms that were previously only studied in an academic setting have started to be adopted in "real world" scenarios. For example, in the field of healthcare, gradient boosting algorithms have been used to predict the likelihood of hospital readmission [3]. In the field of finance, recurrent neural networks have been employed to identify fraudulent credit card transactions [4]. In the field of natural language processing, recurrent neural networks have been utilized for machine translation [5]. Additionally, in the field of computer vision, convolutional neural networks have been applied to object recognition tasks [6]. However, this transition has not been without problems. Academic settings generally assume an application of algorithms in an ideal world, where data is both available and curated. By contrast, applications of machine learning techniques in the "real world" are typically faced with situations where data is either not fully available or, when it is, not particularly well curated.

This lack of quality in the data that is actually available reflects in turn in poorer quality in terms of performance obtained – if any performance can be obtained at all. This in turn affects the rate of adoption of machine learning-based techniques and the trust that is put in such techniques [7]. So, while it can be argued that quality problems are only a limitation of applied machine learning and not of the underlying foundations, it is reasonable to assume that improvements made in this direction will be beneficial to the overall field.

Lack of labelled data is perhaps one of the most interesting scenarios of poor data quality. The interest stems from two specific perspectives:

- *Economic perspective*: a lack of labelled data is typically not due to accidents that occurred during the data collection process (e.g. typos during data entry, malfunctions in sensors). For non-trivial tasks, labels are typically assigned by humans: it follows that labelled data comes at a cost. It is therefore understandable that one may wish to find a reasonable trade-off between a model's performance and the cost of producing the required data.
- *Learning perspective*: the goal of learning from limited labelled dataset goes in the direction of trying to reach an algorithm that mimics the learning capabilities of humans. Indeed, it is a common experience that humans are capable of learning from very limited "training" data. For example, we can identify a new animal species after observing very few pictures of animals belonging to said species. By contrast, typical machine learning algorithms require significant amounts of labelled data to achieve results that are comparable to those of humans.

Given the relevance of the topic of learning with limited label availability, there have been significant advancements from a research perspective. In this section we attempt to provide an overview of the main categories of approaches that can be found in literature. Although far from comprehensive, this overview is meant to provide a context for the topics covered in this work of thesis. Figure 1.1 shows a graphical overview of the scenario. The following paragraphs further elaborate on each of the covered fields.

Unsupervised learning One of the main fields of interest of machine learning, unsupervised learning, is dedicated to the extreme situations where there is no kind of label available whatsoever. The wide variety of unsupervised problems that exist is representative of the importance that "no labels" scenarios have. The common theme of unsupervised learning approaches is that of learning about the structure of the data under study. There are clustering techniques, that focus on grouping data points w.r.t. common behaviors. Similarly, the extraction of frequent patterns attempts to extract, characterize and describe recurring patterns that occur within the data. Additionally, the field of representation learning, which is often carried out in an unsupervised manner, is concerned with the automatic construction of "good" data representations. Among these tasks, we identify the task of clustering as the

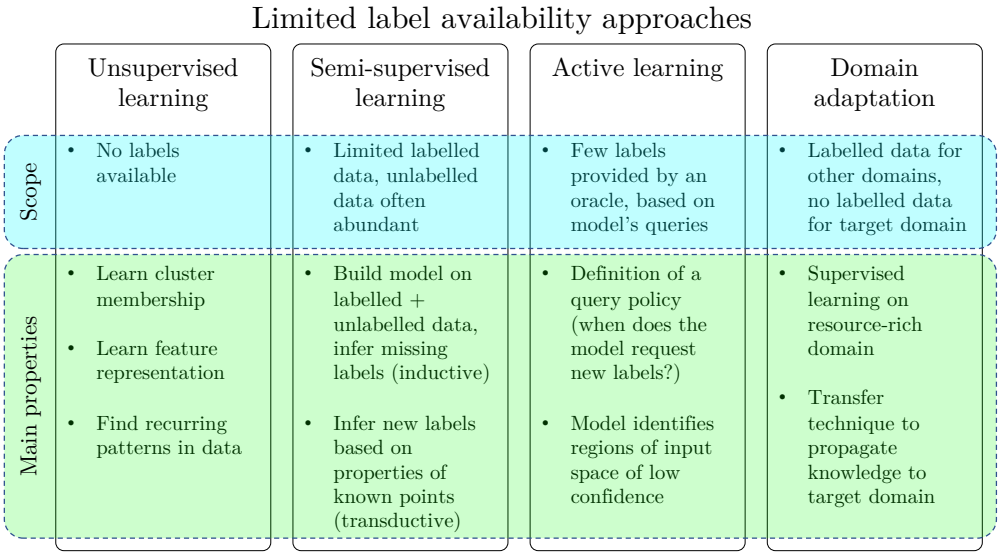


Fig. 1.1 Recap of various limited label availability approaches, defining their scope and their main characteristics.

one being most affine to limited label scenarios, since it allows for the identification of plausible labels.

Semi-supervised learning At the intersection of supervised and unsupervised learning is semi-supervised learning, which is concerned with the learning of supervised models (e.g. classifiers, regressors) starting from large quantities of unlabelled data and a limited set of labelled points. This case well represents the common scenario that arises in limited label availability situations. Semi-supervised learning techniques can be divided into inductive and transductive ones [8]. Inductive techniques focus on learning supervised techniques so as to make predictions even for unlabelled data points (i.e. a model of the input space is learned and then applied to unseen cases), whereas transductive techniques focus on directly inferring unlabelled points' labels based on the properties of labelled points (i.e. without learning a model of the entire input space). These techniques can be often seen as extensions of supervised or unsupervised techniques.

Active learning To reduce the amount labelled data required, active learning algorithms are allowed to query an "oracle" (e.g. a human annotator) to request the ground truth for an ideally small set of points. The idea behind active learning is to

let the learning algorithms autonomously label instances that are well-understood (e.g. those associated with a high confidence of the model) and to request further support in areas of the input space that are not as well understood. The result of such techniques are models that can choose the subset of data points from which to learn in a supervised manner [9].

Domain adaptation There are situations where labelled data is actually available in abundance, but for domains that are different from the one we may be interested in. A very common example can be found in the field of Natural Language Processing (NLP). Large amounts of labelled data may be available for resource-rich languages (e.g. English), whereas we may be interested in working on languages that do not have as many resources available. In domain adaptation, a subfield of transfer learning, the goal is to be able to transfer the knowledge that can be learned from a label-rich domain (source) to a low-resource domain (target). If the only labelled data available belongs to the source domain, we refer to the problem as an unsupervised domain adaptation one [10]. Another interesting distinction can be made in terms of numerosity of source domains: some approaches can learn from a single source domain, whereas others can be generalized to handle multiple source domains [11].

1.1 Main contributions

All of the above-mentioned areas of research have been thriving in recent years, a clear indication that label scarcity is being recognized as a significant challenge and limitation to current state of the art techniques. Within this context, this work of thesis is aimed at approaching a subset of the above fields, providing both theoretical contributions and applied use cases. More specifically, the rest of this thesis is organized into three main chapters. Here we provide an overview of the main contributions that can be found in them.

Chapter 2 discusses the topic of domain transfer, with a focus on the field of NLP and a proposed technique to transfer knowledge acquired from a high-resource language to a low-resource one. Despite the focus on NLP, we explore the extent to which this kind of transfer can be applied to other domains. The chapter is based on the work published in [12], with subsequent extensions that build on top of that. The main contribution of that work consists in proposing a

novel technique for the propagation of information on a graph, based on already available knowledge encoded as vector representations. We show that the proposed methodology can outperform other techniques that are instead based on a less flexible graph propagation technique.

Chapter 3 instead focuses on the topic of unsupervised learning. Finding structure within data – without any kind of supervision – is arguably one of the most important aims of machine learning. In particular, we focus on the clustering task, because of its affinity with label-scarce tasks. We acknowledge that the time and memory complexities of many clustering algorithms (e.g. hierarchical agglomerative clustering [13], DBSCAN [14]) are often prohibitive, $O(n^2)$ or higher. Even when $O(n)$, the constant coefficients may make some algorithms excessively expensive, or their application unfeasible for streaming data. We thus focus on Self-Organizing Maps [15], a neural network-based technique with linear and incremental learning, and propose a training technique that significantly reduces the overall time, at little cost in terms of performance. This technique is based on the initial training of a smaller model, with a propagation of the learned information to a larger one during a tuning phase. We show that the proposed methodology achieves a significant improvement in terms of training speed, with a small impact as far as performance is concerned.

Chapter 4 addressed the topic of semi-supervised learning. This topic is of fundamental importance when learning in limited labels availability scenarios. The assumption of labelled data scarcity does not typically stem from an overall lack of data, but rather from the significant cost of the human labor required for the labelling process. Thus, the main assumption made in semi-supervised scenarios (i.e. that only limited labelled data is available, but there is a wide availability of unlabelled data) is a reasonable one. In this chapter we will discuss how a recently proposed state-of-the-art technique works to learn from both supervised and unsupervised data, by means of pseudo-labels and weak/strong data augmentations. We propose building a more robust confidence detection mechanism for the model, so as to better assess whether a pseudo-labelled point could be a useful addition to the training set. We present some preliminary results that show that the proposed confidence estimation technique outperforms the original model, especially at early training stages, although we acknowledge that significant work still needs to be done in that direction to draw significant conclusions.

Chapter 5 covers an applied use case in the field of predictive maintenance. When the phenomenon of interest is one that is rarely observed (e.g. failures of a system) it is common to have few, sometimes unreliable labels that describe the problem. It is for these reasons that we approached a first predictive maintenance task by applying both semi-supervised and unsupervised techniques on top of traditionally supervised ones. This chapter is mainly based on work published in [16] and [17]. For completeness, we additionally cover some of the differences with a later work [18], focused on a different predictive maintenance task, where the experimental design has been driven by the previous experience.

Because of the multi-faceted nature of this work of thesis, each chapter is mostly self-contained. An introductory overview of the problem and of the proposed solutions are provided, with an exploration of the relevant related works. We then introduce the main contributions that have been produced as a part of this thesis. An experimental section is included for each chapter, where the proposed contributions are evaluated in terms of various metrics (e.g. quality of the results, execution time) and compared against other state-of-the-art methodologies. Based on the results obtained, we draw conclusions on the usefulness of the proposed methodologies. Each chapter is closed by a discussion section, where we explore the implications of the proposed methodologies and identify what some of the possible future directions might be. These future directions are identified both as ways of building on top of the proposed methodologies, as well as exploring completely new directions of interest.

Following the four main chapters of this work of thesis is Chapter 6, which draws conclusions based on the work proposed in the previous chapters. These conclusions will consider both what has been achieved, and what might be relevant for the years to come as far as limited labelled scenarios go.

Chapter 2

Cross-domain knowledge propagation

As already argued in Chapter 1, domain adaptation is a technique commonly used when labelled data is available in a domain that is adjacent to the one of interest. Because of this richness in labels, supervised models can be learned for the adjacent domain. The knowledge learned in this way is then transferred in some way to the domain of interest.

We typically refer to domain adaptation tasks to situations where we preserve the original "building blocks" (e.g. "words" for NLP tasks, "pixels" for computer vision) but we change the underlying domain (e.g. from social media posts to newspaper articles for NLP, of from photographs to sketches for computer vision).

Other tasks, where the building blocks change between the source and the target domains (e.g. words in English and words in Italian) require completely different approaches. This kind of tasks has been referred to as "extreme adaptation" [19]. Although this kind of task is generally harder than the "shared building blocks" ones, we recognize that many limited labels scenarios can belong to it.

As mentioned, NLP offers various great challenges for this extreme adaptation. In this chapter we discuss a possible approach, contextualized in a sentiment analysis (or opinion mining) task. We then discuss how such an approach can be generalized to other challenges, both within and outside of NLP.

The decision of focusing on a sentiment analysis task stems from the fact that, in recent years, an increasing amount of opinionated data has been recorded in digital forms (e.g., reviews, tweets, blogs). This has fostered the joint use of NLP and ML

techniques to extract people's opinions, sentiments, emotions, and attitude from text, i.e., sentiment analysis problems [20].

Recently proposed approaches (e.g., [21, 2, 22]) aim to predict the sentiment polarity of the analyzed text by means of deep learning techniques. However, these models require large corpus of labelled data in order to train accurate sentiment predictors [23]. Meeting this requirement can be challenging while coping with multilingual and cross-domain data. In particular, the majority of the annotated text is written in English whereas small amounts of data are available for less commonly spoken languages. Hence, tailoring deep learning models to the right language is crucial for developing accurate and portable sentiment analyzers.

A promising strategy to overcome the lack of labelled multilingual training data has recently been proposed by [2]. They propose an approach to propagate sentiment information, encoded into high-dimensional embedding vectors [24], across languages.

The idea behind this is to build, with a supervised approach, sentiment embeddings for the source language (usually in English). Then, using a pre-built lexicon that maps English words to those in the target language, the embeddings for the target language are induced using stochastic gradient descent on a graph that connects source and target words. A more detailed description of [2] is given in Section 2.2. The quality of the sentiment embeddings propagation strongly depends on the richness of the bilingual lexicon used. When a bilingual lexicon is either not available or partly incomplete, the induction phase is unable to effectively propagate the sentiment polarity scores from the original language to the target one.

The main focus of this chapter is to improve the quality of the sentiment propagation phase across languages. We acknowledge that enriching languages that are scarce in labelled contents is beneficial for a multitude of downstream tasks and reduces the gap that currently exists between label-rich and label-poor languages.

The key idea is to enrich lexicon information in the propagation phase by deriving the semantic links among pairs of words from an aligned bilingual vector space. This allows us to exploit the underlying text similarities that are not made explicit in the bilingual lexicon. We use an established model for vector representation of words, i.e., fastText [25]. A learning approach to generate aligned fastText word vectors has recently been proposed [26]. Once trained, the bilingual vector spaces

not only embed lexicon information but also allow us to derive non-trivial semantic text relationships directly from the latent space.

This simplifies the procedure of cross-lingual induction and exploits the vector representation of text in the latent space to infer missing word relationships. The authors of [26] have also published the pre-trained aligned vectors for a large number of languages. Hence, a promptly usable, general-purpose vector representation of text is currently available.

The most significant results presented in this chapter have been published in [12]. In this thesis we additionally cover experiments for what concerns the comparison with deep learning methodologies.

The most interesting contribution of the work presented in this chapter is the achievement of the construction of representations of new "building blocks" (i.e. vectors for words in a new language) that only weakly relies on supervised data, as opposed to the heavily supervised approach proposed by [2]. We further discuss various implications of the proposed approach in Section 2.5. Of similar interest is the fact that the proposed embeddings outperform the state-of-the-art ones by [2] on various multilingual benchmark datasets: as an example, when using the proposed sentiment embeddings with an SVM classifier, we achieve a 10% average macro F_1 score improvement w.r.t. [2] on two Italian datasets.

2.1 Related works

To predict the sentiment polarity of textual reviews, news, and posts, several deep learning-based sentiment analysis approaches have been proposed. Most of them (e.g., [27–29]) are language- or domain-specific, i.e., they are specifically tailored to a given context (e.g., movie reviews, Twitter posts) and language. Hence, model learning assumes that a large enough training set is available. As it has already been argued, this kind of approaches cannot be applied to languages (or domains) where only limited labelled data is available.

To extend the applicability of existing sentiment analysis solutions towards other languages, the use of automated machine translation tools has been investigated [30–33].

These approaches, however, rely on the quality of the machine translation tools available. The machine translation task is typically a strongly supervised one, where the same training corpus is available in both the source and the target language. However, for low-resource languages such availability cannot always be guaranteed. Since the focus of this thesis is on label-scarce scenarios, we strive to identify a solution that does not rely on this kind of labelled data.

Parallel strategies entail (i) building sentiment lexicons tailored to different languages and domains of interest and exploiting them to train supervised models [34] and (ii) integrate syntax-based rules in unsupervised models [35]. However, all the aforementioned approaches require a significant human effort, which has already been accomplished only for the major languages and the most popular domains: once again, low-resource languages are heavily penalized.

To propagate sentiment information across different languages and domains, a deep learning approach has recently been presented [2]. They consider an initial vocabulary of English words for which sentiment embeddings are known and a translation lexicon is given, representing semantic relationships between pairs of words (both non-English and English words) such as translations, synonyms, orthographic variants, and other semantic, morphological, and etymological word relationships. In [2] links between words are extracted from a multilingual Wiktionary dump [36]. However, this requires building a new lexicon for each pair of languages of interest, assuming that enough data is available to be able to work in that direction. Furthermore, some relevant semantic links could be missing in the input lexicon. Our proposal builds on top of the work proposed in [2] and relies on an aligned bilingual vector space, e.g., [26], from which explicit and implicit semantic relationships among words can be inferred. In the following section we summarize the approach proposed in [2].

2.2 Sentiment Propagation Based on Translation Lexicon

To propagate sentiment information to various languages, the approach proposed by [2] generates a sentiment embedding vector v_x for each word x in a multilingual vocabulary V. A sentiment embedding is a high-dimensional vector reflecting the

distribution of the word's sentiment polarities across a large range of domains[1]. If the same word is used in multiple languages, each instance is treated as a distinct word in V.

The sentiment vector associated with each word x in the initial vocabulary V_0 (which only includes word in the source domain) is derived via transfer learning. Specifically, for each domain d_j a linear Support Vector Machine classifier [37] $M_j(x) = w_j \cdot x + b$ is trained from a set of domain-specific textual documents. The classifier assigns a polarity to each word, denoting whether the word is peculiar to the domain under analysis. The j^{th} component of the embedding vector v_x incorporates the coefficient w_j of the linear model M_j. Hereafter, we will assume sentiment information to be known for an initial vocabulary $V_0 \subset V$, which usually consists of a subset of English words (i.e., the most popular language used in electronic documents).

The translation lexicon T_L is a set of triplets $\{(x_1, x'_1, w_1), \ldots, (x_m, x'_m, w_m)\}$ $[x_1, x'_1, \ldots, x_m, x'_m \in V]$ providing evidence of the semantic relationships holding between pairs of words in the multilingual vocabulary. The lexicon maps words in the original language to the corresponding translations. Notice that each word may have multiple translations. To incorporate relationships such as synonyms, orthographic variants, and etymological connections, the lexicon includes also links between pairs of words of the same language. In [2] the translation lexicon is extracted from a multilingual Wiktionary dump [36]. The weight w_q associated with a triplet (x_q, x'_q, w_q) denotes the relevance of the semantic relationship. In [36] it indicates the number of semantic links occurring in the data source. For convenience, we will refer to T_L^x as the subset of T_L where word x appears[2], i.e. $T_L^x = \{(a, b, w) \in T_L : a = x\}$.

The sentiment vectors of all the words in V are populated by propagating the cross-domain polarity scores $\tilde{v}_x$ of the words in V_0 via an iterative optimization approach, i.e., Stochastic Gradient Descent (SGD). The optimization problem addressed by the SGD entails assigning values to the sentiment vector v_x for all words x in the multilingual vocabulary V according to the following objective function:

[1]The vectors used in the experiments have 26 dimensions, one for each of 25 domains used by the authors plus an extra dimension combining all domains together.

[2]Since the lexicon represents an undirected graph, we assume that all triplets in T_L^x are in the form (x, b, w).

$$C \cdot \sum_{x \in V_0} ||v_x - \tilde{v}_x||_2 + \\ - \sum_{x \in V} v_x^T \left[\frac{1}{\sum_{(a,b,w) \in T_L^x} w} \cdot \sum_{(x,b,w) \in T_L^x} w \, v_b \right] \tag{2.1}$$

where, given a word $x \in V_0$, $\tilde{v}_x$ represents its initial sentiment vector (learned through transfer learning).

The first term of the loss function ensures that the sentiment vectors of the words in the initial vocabulary V_0 do not diverge significantly from the original ones, for a large enough constant C. The second term guarantees that the inferred sentiment vectors of words that are linked together in the translation lexicon are kept similar. The two terms of the loss function leverage different distance metrics. This preserves the same magnitude for words in V_0 (first term) and may assign different magnitudes to the words in $V \setminus V_0$: indeed, larger magnitudes could indefinitely decrease the loss function. As a part of the proposed approach, we will also address this aspect.

2.3 Proposed propagation algorithm

In this section we discuss the main contributions of this chapter. The core contributions are published in [12]. In short, the main idea behind this work is to link the semantically related words indirectly according to their similarity in a bilingual word embedding space, instead of relying on a lexicon. We show that this improves the quality of the cross-lingual propagation phase (by projecting polarity scores extracted from a richer text representation based on latent spaces) and reduces the need for labelled data.

2.3.1 Bilingual embedding space

Each word in a dictionary is mapped to a vector in the latent space. The application of word embeddings to address many natural language processing tasks is well established. A pioneering work in this field is the Word2Vec model [24]. fastText is a famous extension of Word2Vec, which has been presented in [25]. fastText provides a more effective vector representation by incorporating sub-words in the

input dictionary. The vectors associated with the sub-words can be conveniently combined in order to generate the embeddings of new words that are not present in the dictionary.

Vector representations of text are generated, separately for each language, using neural network-based architectures. Pre-trained vector models (learned from a Wikipedia corpus) are also available for a large number of languages[3]. To links words in different languages, the per-language models need to be aligned first. The procedure to align bilingual fastText vector spaces is thoroughly described in [26]: in short, it consists in identifying a transformation that can be applied to one of the two embeddings to be aligned. Notably, a large number of pre-trained aligned models is available[4]. This allows users to exploit the general-purpose, multilingual vectors (characterized by 300 dimensions and trained from Wikipedia for 44 languages) without the need for retraining them from scratch.

Let E_O be the fastText embedding space in the original language (e.g., English) and let E_T be the aligned embedding space in the target language (i.e., a language other than English). Each word x in the original language has a corresponding vector v_x^{eo} in E_O. Thanks to the aligned bilingual vector space, we can project v_x^{eo} to the target vector space in order to get the corresponding target vector v_x^{et} in E_T. Notice that the new vector does not necessarily correspond to any real word in the target language.

2.3.2 Sentiment propagation strategy

We exploit word similarities in the latent space to propagate sentiment information. Specifically, let v_x be the sentiment vector of an arbitrary word $x \in V_0$. We assume that these vectors are available (e.g. they have been learned from labelled sentiment data for the original language). In [2] various initialization techniques are used – we focus on using vectors extracted from training linear SVM models. We aim to propagate sentiment information to other words in $V \backslash V_0$. This issue is addressed in two steps: (i) first, we create a word graph representing the most significant pairwise word similarities. (ii) Next, we propagate the sentiment scores to the other words using gradient descent.

[3] https://fasttext.cc/

[4] https://fasttext.cc/docs/en/aligned-vectors.html

Notice that step (i) allows us to get a richer word representation compared to a bilingual lexicon.

Word Graph Creation We use a notion of word graph similar to the one already presented in [2]. In particular, we assume the word graph $\mathscr{G} = (V, E)$ to be an undirected weighted graph connecting pairs of words in V. Edges in E are triples $(a,\ b,\ w)$, where $a, b \in V$ are the connected vertices and w is the edge weight. Words are connected based on whether they share similar meaning (e.g. translations, synonyms).

Differently from [2], we do not rely on a pre-built lexicon T_L to obtain E. Instead, we leverage aligned word embeddings. For each word $x \in V_0$ we explore the neighborhood of vector v_x^{et} in the target latent space to look for the neighbor words that are most semantically related to x. More specifically, we select the K nearest vectors (where K is a user-specified hyperparameter) corresponding to the words of the target languages that are closest to v_x^{et}.

Given the set NN_x of x's nearest neighbors, we create a weighted edge $e \in E$ connecting every $x' \in NN_x$ to x. The weight of the edge connecting words x and x' indicates the pairwise word similarity in the latent space and is computed using the cosine similarity [37]. To avoid introducing unreliable word relationships and to limit graph connectedness, we filter out the edges (links) with weight below a given (user-specified) threshold α. The effect of parameters K and α on the performance and complexity of the proposed approach will be discussed in Section 2.4.

We note that by only choosing the nearest neighbors in the target embedding space, we only focus on identifying the "translations" of words in V_0. We could additionally include other types of relationships (e.g. synonyms) by also including the nearest neighbors in the source domain. Due to the considerations that we will make about the desired sparsity of the word graph, we currently avoid adding this information, but we acknowledge that it could be an interesting future development.

Example. Suppose that the original language is English and the target language is Italian. Let us consider the following input parameters: $K = 5$ and $\alpha = 0.4$. Table 2.1 and Figure 2.1 report an example related to the English word *excellent*. Specifically, Table 2.1 reports the five nearest neighbors of *excellent* while Figure 2.1 shows the word sub-graph associated with that word. Only the first two neighbors of *excellent* are characterized by a cosine similarity greater than or equal to 0.4. Hence, only the

x	***x*'s nearest neighbors**	**Cosine similarity**
excellent	eccellente	0.575
	ottimo	0.513
	apprezzabile	0.369
	buon	0.367
	adatto	0.322

Table 2.1 Example: K nearest neighbors of *excellent*. Original language = English, target language = Italian, $K = 5$

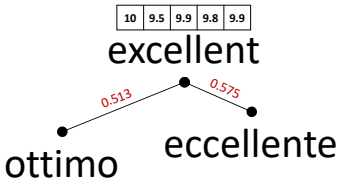


Fig. 2.1 Example: word sub-graph associated with *excellent*. Original language = English, target language = Italian, $K = 5$, $\alpha = 0.4$

Italian words *eccellente* and *ottimo* are connected to *excellent* in the word graph $\mathscr{G}$. This is semantically correct because *eccellente* is the Italian translation of *excellent* and *ottimo* has a similar meaning. The three discarded neighbors are other "positive" adjectives that do not have the same meaning of excellent (the translations of the other three neighbors are appreciable, good, and suitable, respectively). Hence, the enforcement of the minimum similarity threshold helps removing noisy connections.

The English word *excellent*, which is one of the words in V_0, is characterized by a specific sentiment embedding (i.e., a vector of cross-domain polarity scores). The sentiment vectors of the Italian words are populated by propagating the cross-domain polarity scores of the English words via the iterative optimization approach described in the following paragraph.

Gradient Descent with updated loss function The Gradient Descent is used to propagate sentiment information through the word graph. As discussed in Section 2.2, the iterative propagation process should both preserve the values of the vectors in the initial vocabulary V_0 and guarantee a high degree of similarity between the sentiment vectors of linked words. For convenience, we describe the subset of edges that

contain word x as E^x, defined as $\{(a,b,w) \in E : a = x\}$. To achieve these goals, we adopt the following objective function:

$$C \cdot \sum_{x \in V_0} ||v_x - \tilde{v}_x||_2 + \\ - \sum_{x \in V} \left\| v_x - \left[\frac{1}{\sum_{(a,b,w) \in E^x} w} \cdot \sum_{(x,b,w) \in E^x} w \; v_b \right] \right\|_2 \tag{2.2}$$

where, for each word x in the initial vocabulary V_0, the first term minimizes the deviation from its initial sentiment embedding vector $\tilde{v}_x$. The second term minimizes the deviation from the sentiment vectors of neighbors, represented as connected words in the word graph. Adopting the L2-norm in the second term allows the propagation of the vectors information without altering the vectors magnitude. Therefore, words in the initial vocabulary keep, to a good approximation, the same original vectors, whereas new words get sentiment polarity scores similar to those of their neighbors in the target latent space. The distance used for the second term in the proposed loss (Equation 2.2) differs from the one originally used, shown in Equation 2.1. We choose to use an L2 norm for consistency with the first term of the loss. By doing so, we prevent the model from infinitely decreasing the loss by increasing the magnitude of the vectors learned, as it would happen if we attempted to maximize the cosine similarity.

After the gradient descent optimization, the words graph contains words for which a vector representation has been learned. These vectors share the property of being similar to those of their neighbors.

2.4 Experimental results

The experiments presented in this section are aimed at evaluating the quality of the sentiment vectors resulting from the application of the proposed methodology. The evaluation process is formulated as a binary sentiment analysis problem. The sentiment embeddings are compared, in terms of macro-F_1 score, with those produced by the method presented in [2].

All the experiments were run on a machine equipped with Intel® Xeon® X5650, 32 GB of RAM and running Ubuntu 18.04.1 LTS.

The rest of this section is organized as follows. Subsection 2.4.1 describes the settings used in the experimental validation as well as the analyzed datasets. Subsection 2.4.2 summarizes the main results. Subsection 2.4.3 discusses the influences of the main parameters of the performance of the proposed approach. Subsection 2.4.4 analyzes the spatial complexity of the proposed approach. Finally, Subsection 2.4.5 makes comparisons in terms of classification performance using deep models.

2.4.1 Experimental setting

To validate the quality of the generated sentiment vectors, we set up a binary sentiment classification task over multilingual datasets. Specifically, we predict the sentiment polarities of a given set of short text snippets labelled as either *positive* or *negative*.

The main comparisons made are based on the adoption of two popular classification models, i.e. Support Vector Machines (SVM) and the Random Forests (RF) [37].

Classifiers are first trained separately on each multilingual training dataset and then applied to the corresponding test set. More specifically, each dataset is split into a training set (80% of the data) used for training the models and for the tuning of the hyper-parameters, and a test set (20%), which is used for the performance evaluation step.

The adopted classifiers are trained on a vector representation of the input text snippets. The vectors associated with each snippet are computed by averaging the values of the vectors of the words included in the snippet. Other techniques (e.g. using Recurrent or Convolutional Neural Networks [38]) may also be adopted to produce a vector representation that preserves the sequential nature of the input. We defer the comparison of the performance obtained with deep models to Subsection 2.4.5.

The hyperparameters of the classifiers are identified according to the outcomes of a grid search based on a 5-fold Cross-Validation. Separately for each language and dataset, we evaluate the performance of each classification model in terms of macro-F_1 score. The F_1 score is a popular metric that indicates the harmonic mean of precision and recall of the generated model [37]. The macro-F_1 score is obtained as the unweighted average of the F_1 scores for all classes. This provides an overall

Dataset	Cardinality	#Positive	#Negative	% Negative
cs	2,458	1,660	798	32.47
de	2,407	1,839	568	23.60
es	2,951	2,367	584	19.79
fr	3,912	2,080	1,832	46.83
it	3,559	2,867	692	19.44
nl	1,892	1,232	660	34.88
ru	3,414	2,500	914	26.77

Table 2.2 Cardinality and class distribution for each of the datasets presented in [2]

performance estimate that does not penalize minority classes, as would be the case with, for example, the accuracy.

We use as the main competitor the sentiment vectors extracted with the methodology proposed in [2].

Source domain The language used for the source domain is English, as it is the language for which we typically expect to have large quantities of labelled data. The data used for training the source sentiment vectors has been collected from Amazon reviews for products belonging to 25 separate categories, for a total of 1.4 million reviews. These reviews have been assigned a rating of either 1 or 2 stars (for negative reviews) and 4 or 5 stars (for positive reviews). From these reviews, 26 linear models have been trained on a binary classification task: one model has been trained for each of the categories, plus an additional model trained on all the available data. For each word the coefficients learned by each model have been concatenated to produce a 26-dimensional representation that encodes that word's sentiment in various contexts (i.e. categories). The vector extraction process is described in more detail in [2].

Data The list of datasets used for the experiments includes all the datasets adopted by [2], which are described in Table 2.2.

These datasets have been extracted from several websites that collect the reviews left by users on specific topics (e.g. places, movies, food). The target binary class (*positive* or *negative*) is derived from the user ratings. User ratings are not necessarily binary values and generally follow a 5-star system. We have discretized the 5-star

Dataset	Cardinality	#Positive	#Negative	% Negative
IT_1	10,024	5,012	5,012	50.00
IT_2	13,888	6,942	6,946	50.01

Table 2.3 Key statistics for the new Italian datasets

ratings as follows: reviews with 3 stars are discarded (as they are considered neutral). 1's and 2's are assigned to the negative class, 4's and 5's to the positive one.

The statistics reported in Table 2.2 clearly show a strong class imbalance in the analyzed data: the positive class is the majority one. This may hinder the training of robust classifiers, as the minority class may not be sufficiently represented in the training data, resulting in poor learning by the trained models. To additionally evaluate the performance of the proposed approach on more balanced data we have additionally considered two extra datasets for the Italian language (i.e., the language for which the imbalance ratio of the corresponding datasets is maximal). Table 2.3 describes the two new Italian datasets. Data was extracted from reviews of TripAdvisor[5] users in different Italian cities.

2.4.2 Performance comparison

Table 2.4 summarizes the results obtained on the various datasets. For each dataset, the performance for SVM and RF are reported for both the proposed methodology (denoted as Our method) and for the sentiment embeddings produced by [2] (denoted as Dong and De Melo). The outcomes of the proposed methodology were achieved by setting $K = 5$ and $\alpha = 0.4$. Subsection 2.4.3 discusses the effect of these two hyperparameters on the performance of the proposed method.

The proposed methodology for cross-lingual sentiment propagation performs better than the method proposed by [2] in terms of macro-F_1 score on the majority of the analyzed datasets (Russian reviews are the only exception).

To gain insight into classifiers' performance, we additionally explore the precision and the recall of the model. Table 2.5 reports the macro-precision and macro-recall values (indicating the means of per-class precision and recall values, respectively) [37]. Based on the results obtained, we can conclude that classifier

[5]https://www.tripadvisor.com

Dataset	Our method		Dong and De Melo	
	SVM	RF	SVM	RF
cs	**0.7403**	0.7198	0.7227	0.7297
de	0.6847	**0.6981**	0.6495	0.6756
es	**0.6131**	0.531	0.4451	0.4892
fr	0.7021	**0.7291**	0.6389	0.6764
it	**0.8256**	0.794	0.6805	0.6644
nl	**0.6869**	0.6369	0.5903	0.6022
ru	0.6840	0.6112	**0.7221**	0.7009
IT_1	**0.8439**	0.8424	0.7435	0.7311
IT_2	**0.8441**	0.8427	0.7415	0.7494

Table 2.4 Comparison, in terms of macro-F_1 score, between the embeddings produced by the proposed methodology (Our method) and those generated by [2] (Dong and De Melo)

performance is not biased towards either metric – i.e. the two metrics are rather close to one another. Interestingly, the embeddings produced by [2] show – in some cases – higher precision than the proposed method. However, the lower recall produces worse overall results, as measured by the F_1 score.

2.4.3 Parameter analysis

We also study the effect of setting different values for parameters K and α on the quality of the generated embeddings. To do so, we separately analyze their impact on the macro-F_1 scores achieved by the binary classifiers. Hereafter, for the sake of brevity, we will report only the results achieved on a representative dataset (IT_2). It is the largest and more balanced dataset among all the tested ones. Similar results were achieved on the other datasets.

The parameter K indicates the number of neighbors considered when linking words in the source language to those in the target language. The higher K, the more word relationships are included in the word graph. As a drawback, when K is relatively high, the model may include less relevant or unreliable links. Furthermore, since the connectedness of the graph increases, the complexity of the sentiment propagation process increases as well (see Section 2.4.4).

Figure 2.2 shows how the macro-F_1 score varies as K increases, for $\alpha = 0.4$. The plot highlights a knee in the curve for $K = 5$. This implies that, for the purpose

Dataset	Metric	Our method		Dong and De Melo	
		SVM	RF	SVM	RF
cs	Precision	0.7347	0.7326	0.7203	**0.7547**
	Recall	**0.7593**	0.712	0.7474	0.7177
de	Precision	0.6797	0.7481	0.6563	**0.7735**
	Recall	**0.7372**	0.6766	0.7131	0.6507
es	Precision	0.6111	**0.7747**	0.4010	0.6181
	Recall	**0.6154**	0.5428	0.5	0.5172
fr	Precision	0.7025	**0.7301**	0.6488	0.6784
	Recall	0.7019	**0.7309**	0.6403	0.6760
it	Precision	**0.8494**	0.8168	0.6750	0.8030
	Recall	**0.8071**	0.7765	0.7637	0.6336
nl	Precision	**0.6868**	0.6651	0.6059	0.6491
	Recall	**0.704**	0.6317	0.6162	0.6022
ru	Precision	0.6805	0.6623	0.7151	**0.7362**
	Recall	0.7221	0.6025	**0.7634**	0.6845
IT_1	Precision	**0.8441**	0.8425	0.7442	0.7314
	Recall	**0.8439**	0.8424	0.7436	0.7312
IT_2	Precision	**0.8442**	0.8428	0.7416	0.7495
	Recall	**0.8441**	0.8427	0.7415	0.7495

Table 2.5 Results in terms of macro-precision and macro-recall, for embeddings generated by the proposed methodology (Our method) and with those introduced in [2] (Dong and De Melo)

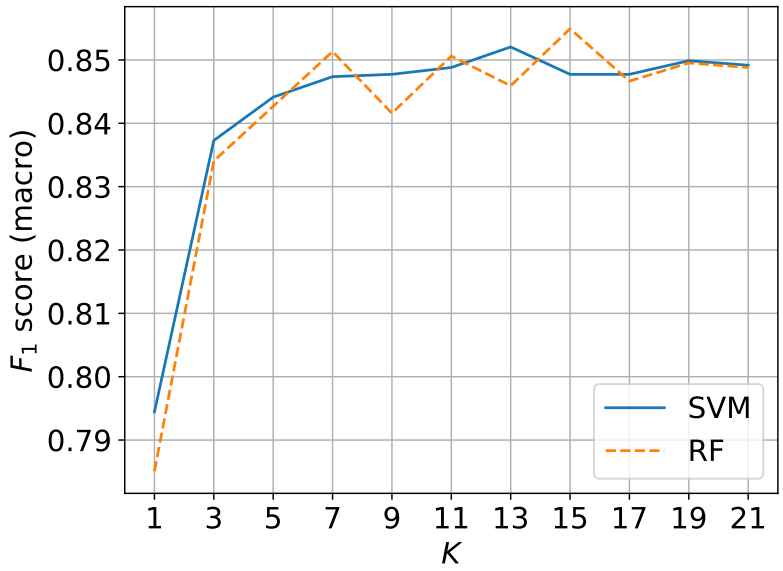


Fig. 2.2 Macro-F_1 score as a function of K, on dataset IT_2

of sentiment classification, using a larger value of K does not yield significant performance improvements. Notice that, to remove the less reliable links, the word graph is pruned early by enforcing the cut-off threshold value α. The impact of the pruning phase is higher when using large values for K.

We separately analyze the impact of the parameter α. Enforcing low α values potentially allows the presence in the graph of "noisy" links, while setting high α values limits the connectedness of the graph. Given an edge $(a,b,w) \in E$, the edge weight w is computed as the cosine similarity between a and b [37]. The cosine similarity takes absolute values between 0 (orthogonal vectors) and 1 (parallel vectors). Figure 2.3 shows how the macro-F_1 score varies as α increases. Due to limitations in the available computational resources, we study the behavior for values of $\alpha \geq 0.4$. The empirical results show that lower values of α produce the best results, with $\alpha = 0.4$ being the best solution we identified due to hardware limitations in terms of memory. When high α values are set (specifically, for values of $\alpha \geq 0.7$), the pruning phase is not beneficial. This means that a noisier graph does not significantly affect the quality of the embeddings learned. This can be explained thanks to the lower importance that these "noisier" connections are assigned. Since we identified limitations in terms of memory, Subsection 2.4.4 provides a more detailed analysis of the space complexity of the problem.

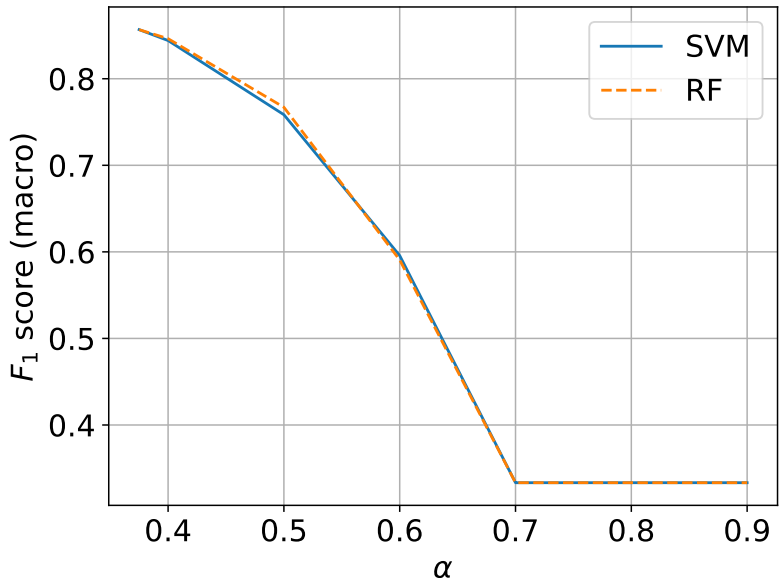


Fig. 2.3 Macro-F_1 score as a function of α, on dataset IT_2

2.4.4 Complexity analysis

The most computationally intensive step of the proposed method is the sentiment propagation on the word graph based on Stochastic Gradient Descent. It entails computing the gradient of the loss function described in Section 2.3 and then iteratively updating the sentiment polarities until a local minimum is reached.

To exploit the hardware optimizations available for matrix computations, the gradient can be computed on the entire matrix, rather than separately for each weight. Specifically, to process the information embedded in the word graph, an adjacency matrix A is defined. Each matrix value A_{ij} indicates the weight of the edge linking two arbitrary words x_i and x_j. If the edge does not exist, the corresponding matrix value is zero. Since graph connectedness is bounded by the cut-off threshold α, the adjacency matrix is rather sparse (the higher α, the sparser A will be). To compute the gradient of the loss function adopted, the adjacency matrix of the word graph is loaded into main memory as a dense matrix. It should be noted that, despite the possibility of representing large sparse matrices efficiently (i.e. by storing only the non-zero elements), the gradient computation requires the dense version to be used.

Let $N = |V_0|$ be the size of the initial vocabulary V_0 in the original language (English, in our case). For each word in V_0, K neighbor words are selected from

the target language. Hence, in the worst case, the word graph contains $(K+1)N$ words. However, since part of the neighbors in the target languages may overlap we can assume, to a good approximation, that each word in the original language has one translation in the target language, yielding $2N$ words in the resulting graph. The corresponding adjacency matrix consists of $4N^2$ cells. Let us assume that B bytes are used to represent each floating point number ($B = 4$ or $B = 8$ in modern systems), the total adjacency matrix size is $4BN^2$. The size N of the initial English vocabulary ranges between $80,000$ and $100,000$ words. Thus, the required memory allocation ranges between 100 and 300 GB (B = 4, N = 80k and B = 9, N = 100k respectively).

A possible way to optimize the process is to identify connected components and to run the Stochastic Gradient Descent on each sub-graph separately. This approach is feasible and exact because nodes and edges outside from each connected components do not influence sentiment propagation within the sub-graph itself. This reduces the size of the processed adjacency matrices, which are stored into main memory: as such, we use this optimization during the gradient descent process. However, as α decreases, fewer – and larger – connected components emerge in the word graph. Therefore, as discussed in Section 2.4.3, in the experimental evaluation reported in this study we have decided to limit the computational complexity of the propagation process by properly setting the K and α parameters.

2.4.5 Deep learning comparison

We additionally tested the quality of the output sentiment embeddings by adopting the same deep learning methodology proposed in the original paper [2]: a Dual-Channel CNN (DC-CNN) model. The architecture is described in detail in the original paper. The main idea behind this architecture is to use two separate convolutional models to process word and sentiment embeddings, through 1D convolutions. The result of these separate models are then merged and used as input of a final fully-connected layer. We adpoted an architecture that replicates the one described in the original paper, including all hyperparameters. A recap of the architecture is shown in Figure 2.4.

Table 2.6 recaps the results obtained, in terms of macro-F_1 score, for the various datasets. The two main contributions of this comparison are in terms of (1) having a 1D convolution applied to address the sequentiality of the data and (2) having word

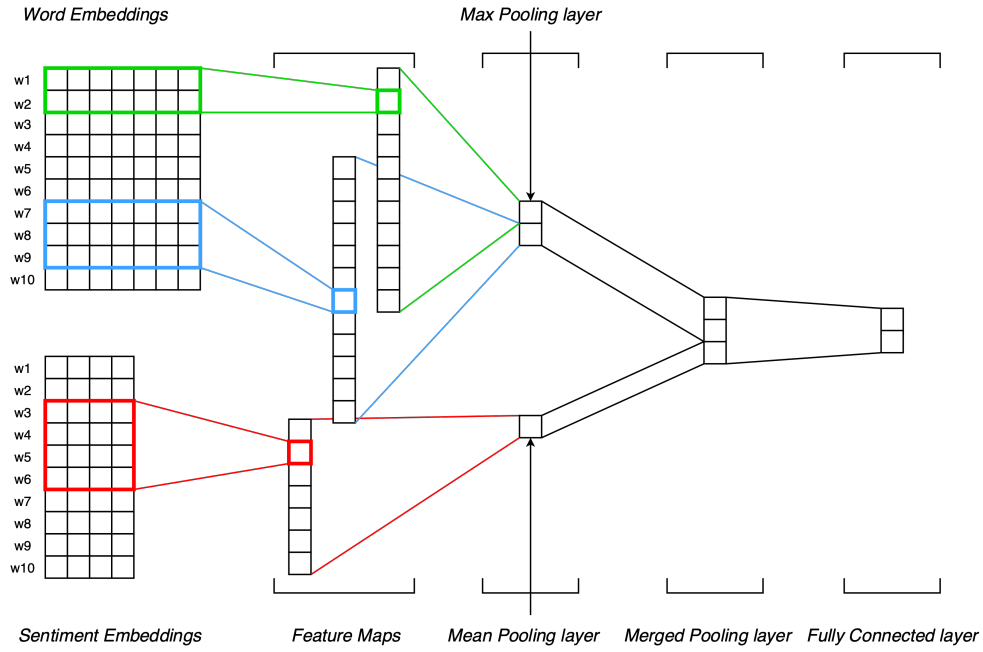


Fig. 2.4 Architecture used for the DC-CNN. The same sentence $\{w1, w2, \ldots, w10\}$ is encoded using both word and sentiment embeddings. 1D convolutions are applied separately. The final, pooled feature maps are then concatenated and fed to a fully-connected layer.

Dataset	Our method		Dong and de Melo	
	DC-CNN	CNN	DC-CNN	CNN
cs	**0.9311**	0.9226	0.9241	0.9149
de	0.8701	**0.9046**	0.8838	0.8874
es	**0.6845**	0.6435	0.6834	0.6611
fr	**0.9168**	0.9078	0.9104	0.8988
it	0.9339	0.9361	**0.9365**	0.9285
nl	0.7087	**0.7352**	0.7195	0.7273
ru	**0.9258**	0.9141	0.8978	0.9187
IT_1	0.9272	**0.9526**	0.9217	0.9471
IT_2	0.9366	**0.9539**	0.9305	**0.9539**

Table 2.6 Results in terms of macro-F_1 score for the proposed methodology and the competitor for the sentiment analysis task solved with a DC-CNN (or CNN in the case of no word embeddings being passed).

embeddings in addition to the sentiment ones. To quantify the extent to which these two changes affect the results, an additional CNN model that only receives as input the sentiment embeddings has been used (labelled as CNN). When comparing the results with the ones in Table 2.4, we immediately notice that, in this case, there is a significant boost in the performance on the task. What is not affected is the fact that the proposed embeddings still outperform the ones from the original work. However, the gap between the two methodologies is not as significant: this is representative of the fact that a more sophisticated model and additional vector representations can provide significant aid. Quite interestingly, the addition of the word embedding information does not always improve the quality of the results obtained and, in some cases, it actually produces worse performance.

Finally, we note that the performance shown in Table 2.6 for the results of the work in [2] have been obtained through a re-implementation of the original architecture. In [2], the only metric for which the results are reported is the accuracy: given the strong class imbalance, we decided to instead focus on the F_1 score, as already argued. Although it is possible to reconstruct some upper and lower boundaries for one classification metric given another one (as we explored in a separate work, [39]), we decided to instead use the same model implementation for both the proposed embeddings and the ones presented in [2]. This is to guarantee that the only variable that is evaluated is the quality of the sentiment embeddings and not other implementation details that are beyond our control.

2.5 Discussion

The original solution proposed in [2] already addressed the "extreme adaptation" task that is learning information across languages. It did so by providing a solution that did not require any kind of labelled data in the target language. However, it still relied on requiring a well-established lexicon that translates words from the source to the target languages. With the proposed solution, instead, we no longer need this kind of lexicon, and instead rely on having word embeddings that have already been aligned.

Indirect lexicon dependency For the sake of completeness, we should note that the alignment process does require having a lexicon that maps source to target words.

As explained in [26], the lexicon is used to learn a transformation that can be applied to the target embedding, so as to make it aligned to the source one. It can therefore be argued that a lexicon is required nonetheless. However, we highlight the fact that the lexicon needed for the alignment does not need to be exhaustive: only a few points can be used to produce a transformation. After the mapping function is applied, we can leverage the transformed embedding to infer more sophisticated relationships between words in the source and transformed target, that where not necessarily known in the original lexicon. Finally, we note that the learning process for both source and target word embeddings only requires unlabelled data – we can therefore learn good embeddings even for languages that do not happen to have large labelled datasets (the full learning approach is described in [40]).

Multiple source and destination domains Both the proposed methodology and the one presented in [2] support having multiple source and destination domains. Our methodology is not concerned with having agreement between the various languages, since all embeddings are aligned to the same vector space. The main limitation that can occur when learning with multiple source/target languages is given by the memory required for the gradient descent process to occur. As already discussed, increasing the number of words and connections affects the scalability of the proposed approach. Although introducing new domains would grow the size of V, we can argue that the proposed solution would not be significantly affected from a practical standpoint. We already discussed that the gradient descent algorithm can be applied to each connected component independently of all others. Adding words from multiple languages is expected to introduce new nodes in the graph but we can expect the connectedness of the "original" (i.e. with only two languages) graph to not change by much, if we assume that words from additional languages roughly belong to a single connected component: this is a reasonable approximation, since words across different languages will still typically connect to the same few words. Thus, we expect the resulting connected components to grow proportionally to the number of new domains added. If instead the new domains consistently "merged" previously disconnected components, we could instead argue that a multi-domain approach would not be as applicable. We have additionally introduced two parameters, K and α, that allow controlling for the connectedness of the graph.

Generalization to other tasks The decision of focusing on the sentiment analysis task is mainly piloted by the already existing works in the field. Within NLP, the same propagation technique can be adopted to propagate information on domain-specific word embeddings, i.e. embeddings that have been fine-tuned to reflect the peculiarities of a specific domain [41] (e.g. scientific writing, or social media posts).

Interestingly, a similar technique can be applied to fields outside of the scope of NLP, when the same entity can be found in different contexts. For example, in graph machine learning, separate graphs may contain the same nodes. For example, a user may have accounts for multiple social network websites, and is thus part of different social network graphs. Existing techniques allow for the learning of node embeddings (e.g. node2vec [42], DeepWalk [43]). The proposed technique could be adapted to allow for the estimation of information for nodes that are not found in one of the two graphs: in other words, we could propagate the information learned on users in one graph to users in the second graph.

Other options could also be available for all those scenarios where we have entities that can be represented with multiple vector representations, across different contexts, and for which we have some kind of information that needs to be propagated.

Chapter 3

Fast training of self-organizing maps

Within the context of limited labels learning, unsupervised learning definitely has a key role, since it is the only field that covers situations where no label whatsoever is available. Instead, algorithms infer common properties found within the data (e.g. frequent patterns). Various techniques approach this task differently. Self-organizing maps (SOMs) are one such technique that offers interpretable results by identifying topological properties in high-dimensional datasets and projecting them on a typically 2-dimensional grid.

More specifically, SOMs are a type of artificial neural network used for unsupervised learning. They consist in a (typically) 2-dimensional grid of nodes, each with its own set of weights. During the training process, these weights are iteratively updated, becoming centroids for the clusters that emerge in the data. This process maintains the dataset's topological properties on the 2D grid. Centroids that are close in the input space will be close on the grid and further away from dissimilar ones.

The purpose of self-organizing maps is to model a possibly high-dimensional problem using a low-dimensional representation. This offers insights on the dataset properties and, by building a 2-dimensional grid, it may be adopted as a useful visualization technique. Additionally, the trained nodes can be used for quantizing the points in the dataset (e.g. for compression, or bucketing purposes).

In recent years, self-organizing maps have been used in a large number of works in a variety of fields (among the others, hydrology [44], physics [45], cytometry [46] and sociology [47]). In particular, all of these works leverage the self-organizing

map's interpretability by extracting 2-dimensional visualizations and making use of the topological properties that self-organizing maps learn.

Self-organizing maps have the interesting property of being one of the few clustering algorithms that (1) is linear in the number of data points available and (2) allows for online learning (i.e. incrementally learning as new data arrives, without necessarily storing the entire dataset at any given time). Because of these properties, SOMs can be used for unsupervised learning of large datasets. This is particularly convenient considering that – in general – it is fairly easy to gain access to large quantities of unlabelled data, since no human-assigned labels are necessary.

Despite the linearity of the training time w.r.t. the number of input points, the constant factor introduced by SOMs may sometimes make it infeasible to learn useful information within a reasonable amount of time. During training, each data point needs to be compared against each of the nodes of the grid. The size of this grid defines how granular the results will be. Hence, a larger self-organizing map typically offers better insights, but at the cost of a higher training time.

Part of this work of thesis has been focused on developing a fast approach to train self-organizing maps. We propose a training algorithm for self-organizing maps that relies on 2 steps. In the first step, a fraction of the available dataset is used to train a smaller SOM. The weights of the model learned this way are then used to initialize the weights of the larger, final SOM. In the second step, the larger SOM is fine-tuned using the remaining fraction of the dataset.

This approach maintains performance comparable with standard self-organizing maps, but at a fraction of the training time. Thus, larger problems that would not have otherwise been tractable, become approachable. We analyze the performance of our approach in detail, by considering several factors. This allows us to both define meaningful values for the required hyperparameters, and understand how the model works in specific situations (e.g. when data is scarce).

3.1 Related works

Self-organizing maps were first introduced in [15]. Originally, weight initialization for the nodes was done randomly and resulted in already well-performing maps. Another established approach is to perform the initialization based on the adoption

of the first principal components. This approach has been shown to have substantial practical advantages over alternative ones [48].

These approaches to initialization, though, introduce a significant computational overhead to the training of the self-organizing map. Further studies focused, instead, on fast initialization techniques that helped SOMs converge more quickly. Among these, [49] adopts the centroids extracted with the k-means algorithm as initial nodes for the map (with a heuristic for the placement of the nodes to preserve some topological order). Then, the weights are fine-tuned with a partial training of the SOM. Another approach, presented in [50], consists in the following three steps: (1) identify – within the dataset – points that are furthest apart from one another (thus building a "hyperbox" that wraps the points in the input space) and use them as initial weights for the corners of the grid, (2) assign the weights of the edges of the grid as an interpolation of the corner weights and (3) fill the rest of the weights through similar interpolations. This approach, as the authors point out, requires $O(M^2)$ comparisons (M being the number of points in the dataset) and is sensitive to outliers. To work around these problems, their solution is once again based on using an initial k-means step to quantize the dataset. These previous works address the same problem we are facing in this work. However, our approach does not rely on k-means nor any other algorithm other than self-organizing maps. The proposed approach will be shown to train significantly faster than k-means: in these terms, the proposed approach is shown to be a better candidate for larger datasets.

There have also been efforts toward building distributed versions of self-organizing maps to cut down the training time. One such work is presented in [51]. The SOM is divided into non-overlapping smaller maps that are distributed across different workers. For each point of the training set, each worker first identifies the local "winning" node (see Subsection 3.2.1 for more details), then all workers are synchronized to identify the global "winner". Finally, with this knowledge, each worker updates its map's weights. The union of all maps results in the final SOM. More recently, other distributed approaches to self-organizing maps have been proposed in [52]. They introduce a Map-Reduce approach where the *map* function is tasked with identifying the winning neuron for each point of the dataset (which is split across multiple workers) and emitting a *(winning node, input point)* key-value pair. Then the *reduce* function performs batch updates for each of the self-organizing map nodes. Both these distributed approaches leverage multiple workers to distribute the training of standard SOMs. Our approach is instead based on a single worker. Since

the proposed algorithm is based on the training of standard SOMs, our approach can be easily distributed across multiple workers using either [51] or [52].

3.2 Fast Self-Organizing Maps Training

This section will first offer a brief formal introduction to self-organizing maps, with considerations on their training complexity, followed by the presentation of the proposed 2-step approach to self-organizing maps training.

3.2.1 Self-Organizing Maps

A self-organizing map is comprised of a set of neurons, or nodes, displaced on a 2-dimensional grid (for simplicity, we assume the grid to be a square of $Q \times Q$ nodes, but all considerations made can be extended to rectangular grids). Let $X \in \mathbb{R}^{M \times N}$ be the available dataset, where each of the M rows represents a point in $\mathbb{R}^N$. Each node n_j of the SOM is associated with a weight vector $w_j \in \mathbb{R}^N$, typically randomly initialized. Additionally, at any step t of the training phase and for each node n_j a set of neighbors $N(n_j,t)$ can be defined (i.e. a set of nodes close to n_j on the grid). The time dependency is necessary to allow for changes in the neighborhoods throughout the training (e.g. by shrinking them during later steps). During the training phase, for each point $x \in X$, the closest node n_c according to a distance (e.g. the Euclidean one) is identified as:

$$n_c = \underset{n_j}{\operatorname{argmin}} \|w_j - x\| \tag{3.1}$$

n_c will also be referred to as the "winning" node. The weights of the nodes in the neighborhood of n_c, $\{w_j : n_j \in N(n_c,t)\}$, are updated to better resemble x:

$$w_j(t+1) = w_j(t) + \alpha(t)\beta_{jc}(t)(x - w_j(t)) \tag{3.2}$$

where $\alpha(t)$ is a learning rate and $\beta_{jc}(t)$ is a coefficient that dampens the learning as the neurons get further away from the winner. By training the neighboring nodes as well as the winning one, a topological order forms among neurons, where nodes that

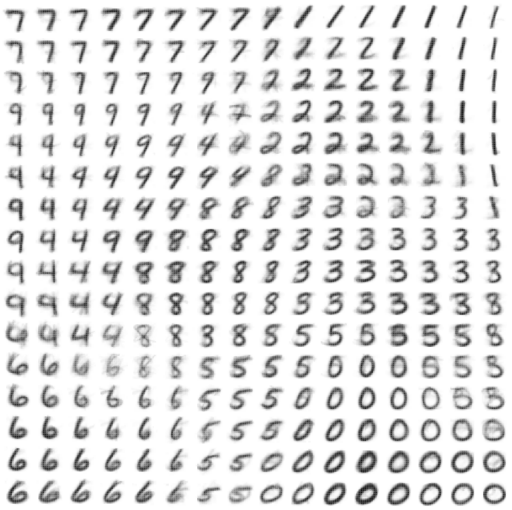


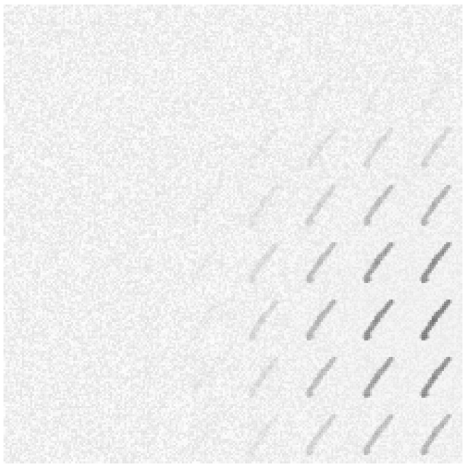
Fig. 3.1 16×16 trained SOM

model similar points are close to one another. An example of a 16×16 SOM trained with the MNIST dataset (see Section 3.3 for more details) is shown in Figure 3.1. Each cell of the grid represents one of the nodes trained by the self-organizing map. The 784 pixels that represent each node (28×28, the resolution of each MNIST digit) are the weights learned. The grid shows how weights have evolved to resemble samples of the dataset, with similar shapes being close to one another. An interesting example of how "similar" nodes end up close to one another is the following. The first row of Figure 3.1 has nodes that resemble the digits "7" and "1". "1"s that present a slant are placed closer to the "7"s because of their greater similarity.

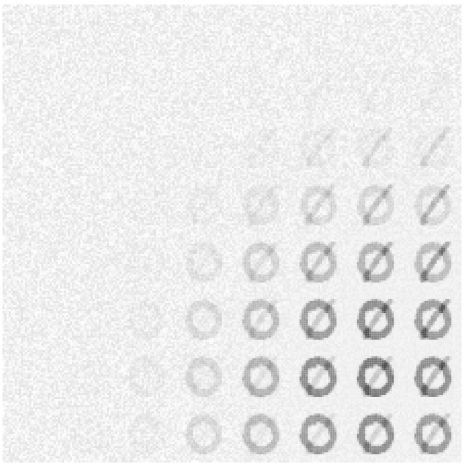
Figure 3.2 shows the first three training steps of a self-organizing map. Figure 3.2a shows the initial random weights. Figure 3.2b shows how the weights are updated after the left-most of the inputs in Figure 3.2e is used for the training. One of the random nodes has a slightly higher similarity to the input. The weights of this node are then updated (based on Equation 3.2) to better resemble the input, along with the weights of its neighbors. The same happens in Figures 3.2c and 3.2d for the second and third inputs.

During the training phase, for each of the points in the dataset, the distance from each node of the grid is computed. For a square $Q \times Q$ grid, MQ^2 operations are performed over the entire training set. This sets an upper bound to the maximum size of the SOM, for a fixed time budget. Since having a larger number of nodes in

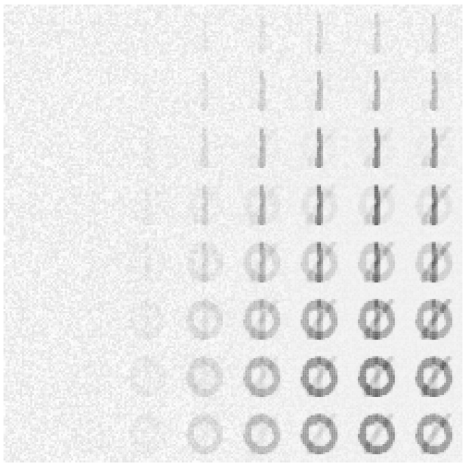
(a) Step 0 (random initialization)



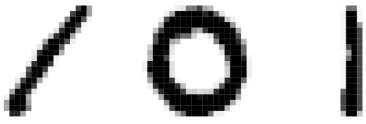
(b) Step 1



(c) Step 2



(d) Step 3



(e) Inputs

Fig. 3.2 First three weight updates during the training of an 8×8 SOM. In (e) are depicted the 3 inputs used for the 3 training steps.

a SOM helps better represent a dataset (and build more meaningful clusters), this limitation hinders the potential adoption of SOMs for processing very large datasets.

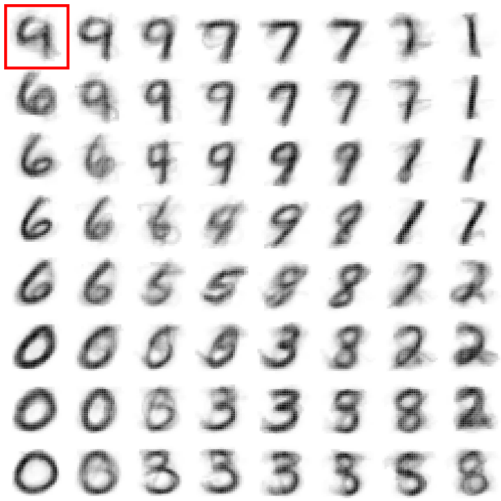
3.2.2 2-Step Training Approach

To reduce the time complexity of SOM training, we propose to split the training phase in two steps. The first consists in training a smaller SOM, to produce a "coarse" grid with a low number of nodes that represents an approximate topology of the dataset. As a second step, this grid is extended to the final SOM size (with a larger number of nodes), to capture local topologies (e.g., sub-clusters that form within any of the coarse clusters). Hence, the initial effort of converging from the entire N-dimensional input space to its meaningful portions is done on a smaller number of nodes, thus requiring a significantly lower computational effort.

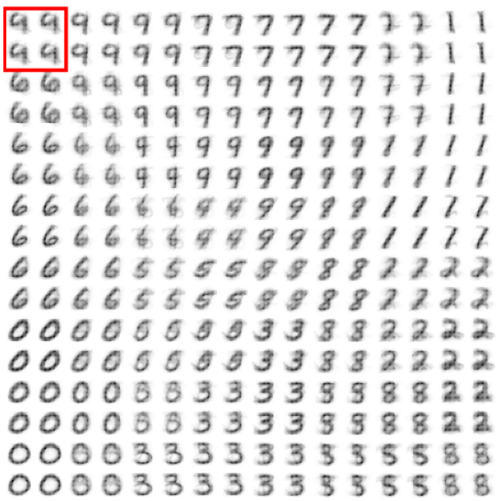
To achieve this model expansion for a $Q \times Q$ map, a single $P \times P$ (where $Q = nP, n \in \mathbb{N}$) SOM has been trained with a fraction η of the original dataset (Figure 3.3a shows one SOM with $P = 8$). Then, the trained nodes are replicated n^2 times in their locality, thus building a $Q \times Q$ SOM (as shown in Figure 3.3b, where $n = 2$ and the replicated nodes are the ones from Figure 3.3a). This SOM is then trained with the remaining $1 - \eta$ portion of the dataset. This fine-tuning process is shown in Figure 3.3c. With this 2-step training, only a fraction of the training requires working on Q^2 nodes.

In this work, we only consider a single training epoch (i.e. each data point is only seen once). This is a necessary constraint for online learning. For smaller datasets the training can span multiple epochs. The proposed methodology can be easily adapted to this scenario, by taking η as being the fraction of the total iterations used for the training of the coarse SOM.

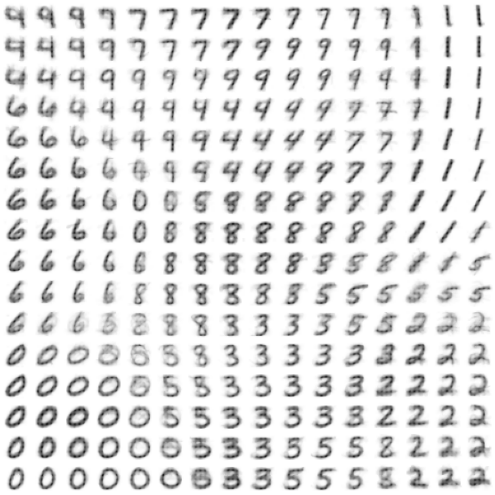
Another assumption made during this work, as explained above, is that $n \in \mathbb{N}$, i.e. the side of the larger SOM is a multiple of the side of the small SOM. This does not necessarily have to be the case, but it does simplify the notation used when explaining the algorithm. In the general case, we could pick any n with the constraint that $Q = nP \in \mathbb{N}$. In that case, the replication process that occurs after the first training step requires some minor adaptation when choosing where to propagate each of the original nodes. Alternatively, the replicated nodes could be obtained as a combination of neighboring nodes (e.g. as with SMOTE [53]).



(a) An 8×8 SOM.



(b) A 16×16 SOM with replicated nodes from Figure (a). In red are the 4 nodes replicated from the single node highlighted in red in Figure (a)



(c) A 16×16 SOM after the fine-tuning of the SOM from Figure (b)

The complexity of training a single $Q \times Q$ SOM, as already stated, is $\propto MQ^2$. Instead, building a single $P \times P$ map with a fraction η of the dataset is $\propto \eta MP^2$. If the small SOM is then replicated n^2 times to build a $Q \times Q$ SOM (with $Q = nP$) and the resulting grid is trained with the remainder of the dataset (i.e. $1-\eta$), the required time is $\propto (1-\eta)Mn^2P^2$, resulting in a total training time $\propto (\eta(1-n^2)+n^2)MP^2$. In terms of complexity, both approaches are $O(MP^2)$, but the proposed approach reduces the training time by a factor ρ, defined as:

$$\rho(n,\eta) = \frac{n^2}{\eta + n^2(1-\eta)} \tag{3.3}$$

Thus, it is evident that $\eta = 0$ is a special case in which the entire dataset is used to train the large SOM and none of it is used to train the small one. In this case $\rho(n,0) = 1$, which is equivalent to training a SOM with the standard approach. The opposite edge case, $\eta = 1$, is the one where the entire dataset is used to train the smaller SOM, resulting in a time gain of n^2. In this case, no fine-tuning is done on the large SOM. In terms of performance (including training time), the resulting SOM is identical to a $P \times P$ one.

Additionally, for a fixed Q, a larger value of n implies a smaller P, thus leading to a faster training of the smaller SOM. The following limit holds:

$$\lim_{n \to +\infty} \rho(n,\eta) = \frac{1}{1-\eta} \tag{3.4}$$

This introduces a boundary on the training time reduction that can be achieved with the proposed approach. This boundary corresponds to the time gain that can be obtained by only training the larger SOM on the smaller fraction of data (i.e., a scenario in which the training time of the smaller SOM is negligible).

3.2.3 Parallel computing considerations

Differently from other pre-training methods presented in the related works, the 2-step approach is based on the training of two standard self-organizing maps, without other time-consuming operations involved. This means that existing works on the parallelized training of SOMs (e.g. [51]) can be applied to the training of the two

SOMs of our approach for a further training time reduction. This reduction is proportional to the number of parallel workers.

We argue that, even when trained on a single worker, the proposed method can achieve competitive training time when compared to other distributed methodologies. In [51], assuming that the training of a $Q \times Q$ SOM is distributed onto m^2 parallel workers, the training time for the entire SOM will be $\propto MQ^2/m^2$, since m^2 smaller SOMs are being trained simultaneously. This reduces the training time of a large SOM down to the time needed to train a SOM that is $1/m^2$ of the original one.

The only additional time the 2-step approach (with $n = m$) requires, compared to the distributed approach, is the time it takes to train a large SOM with a small (20%, for $\eta = 0.8$) fraction of the dataset. Hence the 2-step approach, which runs on a single worker, may reach comparable training times with respect to the distributed approach in [51] requiring m^2 workers.

3.3 Experimental results

In this section, standard SOMs (which will be considered as being the baseline) are compared to self-organizing maps trained with the proposed 2-step approach. The comparisons are based on the following metrics:

- *Training time*: while Equation 3.3 already defines the theoretical time improvement, the actual training time is experimentally measured to verify its accordance with the expected result.
- *Quantization error*: SOMs can be used to quantize a set of points with the trained nodes. Given a dataset X, the quantization error q_e is defined as:

$$q_e = \frac{1}{|X|} \sum_{x \in X} \|x - w_c\| \tag{3.5}$$

  This is an indication of how well the map can represent a given dataset with a limited number of "buckets".

- *Accuracy*: given a labelled dataset, the trained SOM can be used as a classifier. After training, each (labelled) input in the training set is assigned to a node (the winning one for the given point). Each node is then assigned a class label

based on the labels of the points that have been assigned to it (with majority voting). Next, new points can be assigned the label of the node that is closest to them. For this classifier, its accuracy (i.e., the fraction of correctly classified elements) can be computed. We note that other metrics could be used for the evaluation of a cluster assignment based on a known ground truth, for example the Rand index [54]. The Rand index works by identifying all pairs of points that belong to both the same cluster and the same class, or that belong both to different clusters and different classes. However, SOMs can produce multiple clusters (nodes) that all map to the same class (for example, multiple nodes represent each digit in Figure 3.1). Because of this, the Rand index (and metrics that work similarly) would fail to capture the quality of the clustering.

Both quantization error and accuracy have been computed on a separate test set.

For each experiment, three different SOMs have been built: a baseline (i.e. the "standard" SOM) and two SOMs built with the proposed technique, respectively with $n = 2$ and $n = 7$.

The experiments have been performed on a machine running Ubuntu 16.04, equipped with an Intel Xeon X5650 (6 cores, 12 threads) @ 2.66 GHz and 32 GB of memory. The source code has been developed using Python 3.5, with the MiniSom library [55] for the implementation of self-organizing maps and scikit-learn [56] for the comparisons with k-means (an implementation of the optimized algorithm presented in [57] has been used, unless otherwise stated). The main dataset used is MNIST [58], which is comprised of 28×28 hand-written digits from 0 to 9, divided into a training set of 60,000 entries and a test set of 10,000.

The presented experiments will cover various aspects of the proposed approach. First, analyses of the performance as η and Q vary are presented in Subsections 3.3.1 and 3.3.2. Then, Subsection 3.3.3 explores scenarios with smaller datasets. The baseline and the proposed approach are then compared to another clustering technique, k-means, in Subsection 3.3.4. Datasets other than MNIST are used in Subsection 3.3.5 as a validation of the proposed approach and finally, in Subsection 3.3.6, the algorithms are run on larger datasets to assess their scalability.

3.3.1 Effect of varying η

η is the fraction of dataset used to build the small SOM. Hence, $1-\eta$ is used to train the larger SOM.

Two special cases are given:

- $\eta = 0$. In this case, the entire dataset is used to build the large SOM. The training time is expected to be the same as the one for the baseline case.
- $\eta = 1$. In this case, the entire dataset is used to build the small SOM. The training time should be reduced by a factor of n^2 (as only the small SOM is trained). The performance in terms of accuracy and quantization error are expected to be the most degraded, since this corresponds to reducing the SOM size by a factor of n^2 (the final SOM will still have the expected number of nodes, but they will be repeated in groups of n^2).

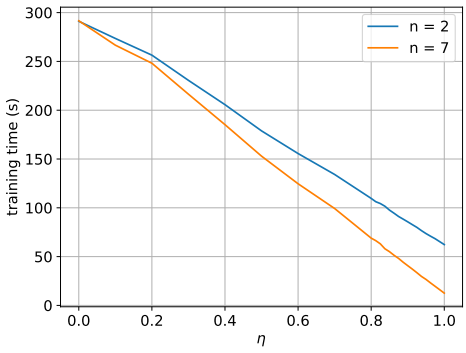
The performance for $0 < \eta < 1$ are expected to be in between these two edge cases. Indeed, Figures 3.4a, 3.4b and 3.4c show this behavior for 28×28 SOMs (this size has been chosen as it presents satisfactory results when trained with the baseline approach, as well as being suitable for $n \in \{2,7\}$).

In terms of training time, the results behave as expected. The case $n = 7$ is faster than $n = 2$. The ratio of the time curves is also in accordance with Equation 3.3, when computing $\rho(7,\eta)/\rho(2,\eta)$ (i.e., the expected ratio of the training times for the two values of n, as a function of η).

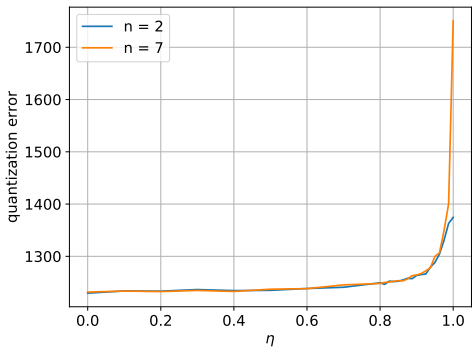
In terms of accuracy and quantization error on the test set, a significant degradation occurs for $\eta > 0.8$. This happens because of the insufficient data for the training of the larger SOM. For this reason, the study of the performance as Q changes have been performed for $\eta = 0.8$ (in this case, based on Equation 3.4, the maximum time gain possible should be 5x).

3.3.2 Effect of varying Q

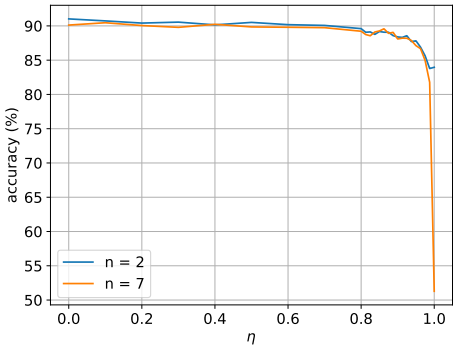
Varying Q alters the shape of the final SOM. It drives the time complexity with a squared factor, making the problem intractable for large values.



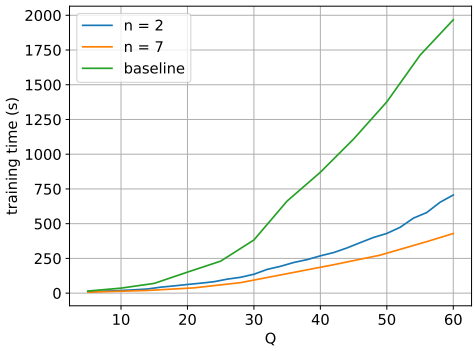
(a) η vs training time



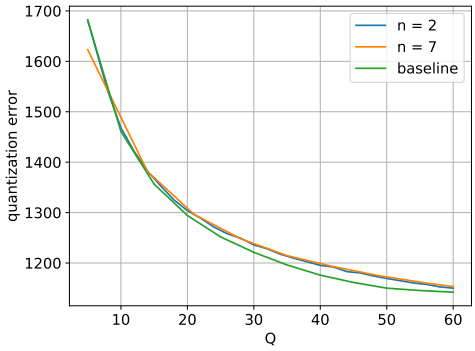
(b) η vs quantization error



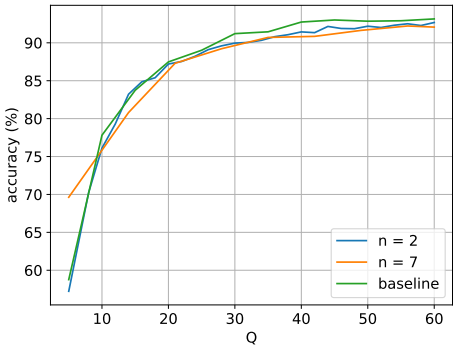
(c) η vs accuracy



(a) Q vs training time



(b) Q vs quantization error



(c) Q vs accuracy

In this analysis Q ranges between 5 and 60. The selection of this range is based on the heuristics adopted in literature, recommending a number of nodes $\approx 5\sqrt{M}$ (M being the cardinality of the dataset) [59]. For MNIST, it corresponds, approximately, to a square map with $Q = 35$. Considering that, as mentioned in [60], a "trial-and-error" approach – if feasible – may identify the best size for the SOM, the range $5 \div 60$ has been used for more exhaustive results.

The most significant result is related to training time, as shown in Figure 3.5a. The reduction in time with the proposed technique is in accordance with the one expected from Equation 3.3. In particular, $\rho(2, 0.8) = 2.5$, and the experimental training time gain is approximately 3x, while $\rho(7, 0.8) \approx 4.6$, the same as the rounded experimental result.

The second significant result involves quantization error (Figure 3.5b) and accuracy (Figure 3.5c). As expected, as Q grows, the quantization error reduces and the accuracy increases. It is particularly interesting, though, that the degradation of the

performance for the proposed SOMs is particularly limited, if not negligible in some cases (on average, the degradation is smaller than 1% of the baseline). Hence, this effect, combined with the lowered training time, yields similar performance with a much lower computational cost or, with the same cost, it allows achieving better performance, by training a larger SOM.

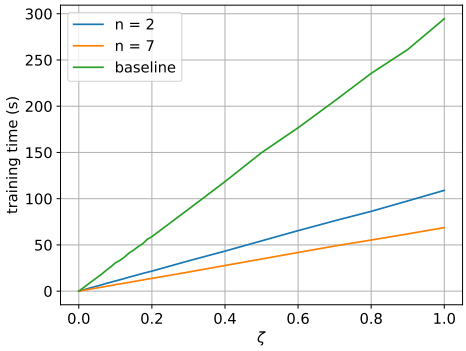
3.3.3 Behavior with low-cardinality datasets

Intuitively, the 2-step approach leverages a large portion of data to "identify" the regions of the high-dimensional space where the data is distributed, displacing the centroids of the small SOM there. Then, the centroids are replicated and fine-tuned with the remainder of the data. An argument could be made that the proposed 2-step approach only works because the fraction $1-\eta$ of data used for the final fine-tuning is large enough to learn the "large" model in its entirety, and that the training of the smaller SOM may not be beneficial to the overall training.

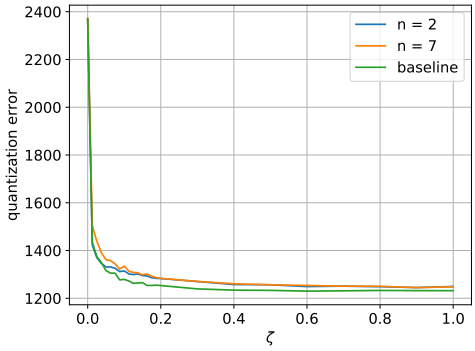
As such, an important scenario to be studied is the one where only limited data is available. To further explore this case, a new kind of experiment has been devised. Here, Q and η have been fixed (to 28 and 0.8 respectively). Then, subsets of the entire dataset have been used to (1) train a standard SOM and (2) apply the proposed 2-step approach. These subsets used are fractions of the original dataset X, ranging from 0% (i.e. an "empty" training set) to 100% (i.e. the entire dataset) of X. Let ζ be the used fraction of X. Then, the proposed approach exploits a fraction $\zeta\eta$ of the dataset for the training of the small SOM, and $\zeta(1-\eta)$ for the large SOM.

The expected result in terms of training time is the same one as before, where the training of the 2-step approach is significantly faster than the baseline approach. In this case, since n and η are fixed, the training time is only linearly dependent on ζ. Indeed, Figure 3.6a shows this linear dependence.

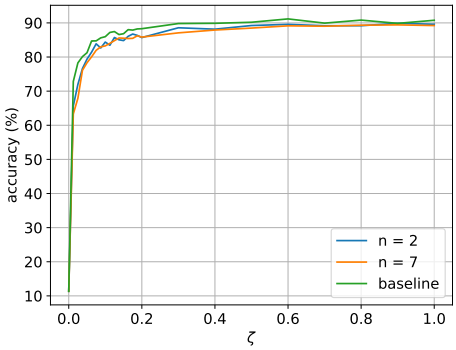
In terms of accuracy and quantization error, the performance of the baseline and of the proposed approach are expected to be similar as ζ varies. A divergence between the curves would imply that the 2-step model only provides satisfactory results because the fraction $1-\eta$ of data used for the final fine-tuning is still sufficient to learn a model from scratch (thus making the proposed approach no better than a random initialization). As shown in Figures 3.6b and 3.6c, though, there only is a slight, constant performance degradation between the two models. This is evidence



(a) ζ vs training time



(b) ζ vs quantization error



(c) ζ vs accuracy

of the fact that there is indeed an advantage in using the 2-step approach that is not only due to the dataset size.

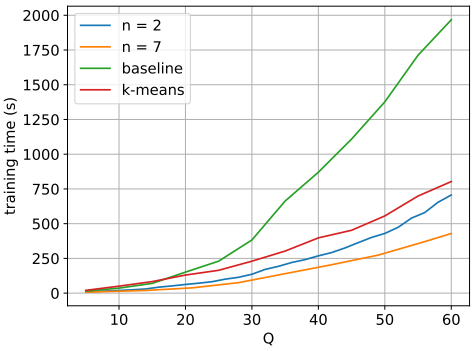
3.3.4 Comparison with k-means

To understand how the proposed SOM approach fares in absolute terms, a performance comparison with other state-of-the-art unsupervised techniques is needed. Of the many unsupervised algorithms, k-means [61] is one of the most relevant for this comparison, because of the many traits it shares with self-organizing maps. Indeed, both algorithms rely on the definition of the number of clusters to be identified (k for k-means, Q^2 for SOMs) and both introduce an iterative approach where centroids are identified by repeated adjustments. The two models differ in that k-means does not allow for online training, since it requires the entire dataset to compute the centroids at each iteration.

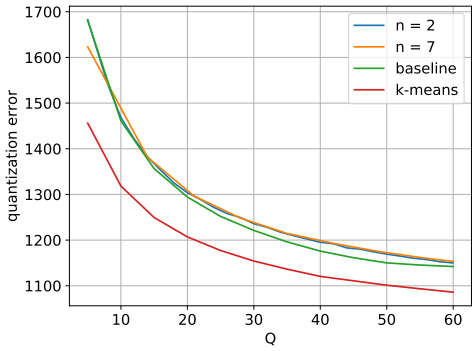
Self-organizing maps have an advantage over other clustering algorithms: The output of SOMs is a 2-dimensional grid in which nodes are distributed based on the similarity among one another. This is not the case with k-means, which instead provides points in a (possibly) high-dimensional space, with no relationships between one another. The output of self-organizing maps proves particularly useful when used in combination with visualization techniques for interpreting the results (instead of reducing the dimensionality of the results through other typically computationally expensive algorithms, such as PCA or t-SNE), or as a pre-processing step where the points are distributed into buckets that have a concept of neighborhood and of distance (e.g. Manhattan) in a low-dimensional space.

Comparisons between various unsupervised algorithms have already been done extensively in literature. One study in particular [62] is focused on comparing self-organizing maps and k-means. The results presented in that work highlight how, on the datasets used for the study, k-means has a slight performance advantage when compared to SOMs. As such, we expect this to also be the case for the proposed 2-step SOM.

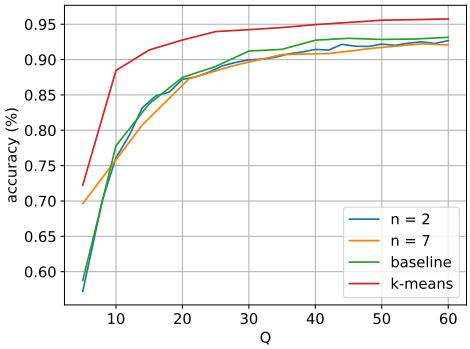
The performance of the three algorithms (k-means, standard and 2-step SOM) are compared in terms of training time, quantization error and accuracy. As expected, k-means performs better than either SOMs in terms of quantization error and accuracy, as shown in Figures 3.7b and 3.7c. In particular, k-means is approximately a constant



(a) Training time for k-means and SOMs



(b) Quantization error for k-means and SOMs



(c) Accuracy for k-means and SOMs

2-3% more accurate than SOMs for larger values of Q, completely in accordance with the results presented in [62].

In terms of training time, instead, two important clarifications needs to be made.

- Most implementations of k-means are executed multiple times, each with a different initialization. This is done to help reduce the impact of local minima solutions. To present a fair comparison, the k-means results presented are all based on a single initialization. The results in terms of accuracy and quantization error do not vary significantly when using multiple initializations, but the training time for T initializations is obviously T times that of a single initialization.
- The adopted k-means implementation parallelizes part of the workload, by distributing it across 12 concurrent workers[1]. The adopted SOM library, on

[1] Because of the 12 cores used for the experiments.

the other hand, runs a single worker. Running k-means without parallelization leads to heavily slower training times when compared to self-organizing maps (so much so that a comparison between the them would be meaningless). Hence, only the parallel training times are reported for k-means.

Figure 3.7a shows the training time for the different algorithms. It is particularly interesting that, with a standard implementation, self-organizing maps are slower than the parallelized version of k-means. This, paired with the slight underperformance in terms of other metrics, made SOMs not competitive with other unsupervised algorithms.

The proposed 2-step approach, on the other hand, is faster than k-means. Additionally, as already mentioned, SOMs introduce a level of interpretability that is missing from k-means and other unsupervised techniques. While this interpretability comes with a slight degradation in performance, this trade-off is still useful for those domains where data mining techniques are used in human-in-the-loop scenarios.

3.3.5 Additional datasets

To validate the proposed technique, the 2-step training approach has been also applied to the following datasets (in addition to MNIST):

- *Fashion-MNIST* [63]. A dataset of fashion items (e.g. shirts, trousers, bags) that maintains the same image size (28×28 grayscale), the same train-test splits (60,000 training points, 10,000 test points) and the same number of classes (ten) as MNIST. This is intended as a more challenging alternative to MNIST.

- *CIFAR-10* [64]. A collection of 32×32 color images divided into 10 categories, with a training set of 50,000 samples and a test set of 10,000.

- *Character Trajectories* [65]. A UCI dataset that contains a collection of multiple labelled samples of pen tip trajectories recorded while writing individual characters. For each character, three time series are collected: the pen tip velocity for the x and y axes, and the pressure applied to the pen tip. The dataset is available in a normalized form, but with various time series lengths (depending on how long it took to draw the character). For uniformity, all

signals have been upsampled to match the longest time series available (205 samples). The training/test split ratio has been set to 6/1 (i.e. the same used for MNIST). The dataset contains the data for 2,858 hand-written characters, each belonging to one of 20 classes. Given the low number of rows for this datasets, all algorithms have iterated 10 times over the training set.

The results obtained and the SOM configurations are displayed in Table 3.1. The results obtained on these additional datasets are consistent with the ones presented for MNIST. A significant (up to 8x for $(\eta, n) = (0.9, 7)$) training time speed-up occurs for the 2-step approach, at the cost of a slight degradation in performance. In terms of accuracy, this degradation is $\approx 1\%$ for all cases, with the exception of Character Trajectories. For that dataset, the performance degradation is slightly more impactful – 2 to 4% less than the baseline case. Considering that the test set for this dataset is comprised of only 409 samples, every misprediction has a more significant impact on the performance degradation compared to the other datasets. Similar considerations can be made in terms of quantization error.

3.3.6 Algorithm scalability

To assess the time gain of the proposed approach on larger scales, two synthetic dataset have been used. These have been generated by sampling various Gaussian distributions in a high-dimensional space. *Dataset 500k* includes 500,000 entries, each in a 500-dimensional space, with a total of 50 separate clusters. *Dataset 1M*, instead, contains 1,000,000 entries in a 1,000-dimensional space, with 100 clusters.

The training times have been measured for $Q \in \{50, 100\}$ for the baseline approach, for k-means and for the 2-step approach with $(\eta, n) = (0.8, 5)$. Considering that all models successfully identified the clusters in the datasets (i.e. all accuracies are 100%), only the results in terms of training time are proposed in Table 3.2.

All 2-step versions have a training speed-up in accordance with $\rho(5, 0.8) \approx 4.3$ (on average, the experimental factor is of 4.5, with a standard deviation of 0.1). For both datasets, the version of k-means that implements [57] runs out of memory before completion. Hence, the version based on [61] has been used instead. This introduces an impactful increase in training time. This makes k-means even slower than standard self-organizing maps: for Dataset 1M and $Q = 100$, the execution of

Dataset	Method	Q	η	n	Training time (s)	Quantization error	Accuracy (%)
MNIST	baseline	28	-	-	323.13	1232.71	90.29
	k-means	28	-	-	202.92	**1162.21**	**94.17**
	2-step	28	0.7	2	136.16	1242.02	89.71
	2-step	28	0.8	2	116.49	1247.04	88.96
	2-step	28	0.9	2	87.95	1262.59	89.02
	2-step	28	0.7	7	103.79	1243.75	89.57
	2-step	28	0.8	7	73.58	1245.86	89.39
	2-step	28	0.9	7	**41.38**	1263.18	89.07
Fashion-MNIST	baseline	28	-	-	312.01	1031.78	77.93
	k-means	28	-	-	178.32	**976.97**	**80.21**
	2-step	28	0.7	2	139.05	1037.45	77.75
	2-step	28	0.8	2	111.13	1042.90	77.51
	2-step	28	0.9	2	87.79	1053.60	77.20
	2-step	28	0.7	7	102.89	1039.47	77.50
	2-step	28	0.8	7	72.62	1044.39	77.98
	2-step	28	0.9	7	**41.71**	1053.46	77.32
CIFAR-10	baseline	28	-	-	1464.72	2394.83	32.88
	k-means	28	-	-	568.33	**2335.88**	**35.16**
	2-step	28	0.7	2	640.77	2402.47	32.20
	2-step	28	0.8	2	516.65	2407.27	32.12
	2-step	28	0.9	2	373.68	2418.99	31.56
	2-step	28	0.7	7	474.76	2403.36	32.12
	2-step	28	0.8	7	322.18	2406.28	32.19
	2-step	28	0.9	7	**171.98**	2421.23	31.94
Character Trajectories	baseline	28	-	-	92.45	5.51	94.87
	k-means	28	-	-	38.20	**5.45**	**97.56**
	2-step	28	0.7	2	42.81	5.93	92.67
	2-step	28	0.8	2	35.44	6.25	91.20
	2-step	28	0.9	2	28.28	6.68	90.71
	2-step	28	0.7	7	30.84	5.94	92.67
	2-step	28	0.8	7	22.36	6.27	92.42
	2-step	28	0.9	7	**13.10**	6.73	90.22

Table 3.1 Performance on various datasets, for the baseline model (SOM), k-means and the proposed 2-step approach, trained with various configurations of Q, η, n.

Dataset	Method	Q	η	n	Training time (hh:mm:ss)
Dataset 500k	baseline	50	-	-	01:50:58
	k-means	50	-	-	04:25:36
	2-step	50	0.8	5	**00:25:14**
Dataset 500k	baseline	100	-	-	08:47:22
	k-means	100	-	-	16:52:35
	2-step	100	0.8	5	**01:54:45**
Dataset 1M	baseline	50	-	-	07:30:49
	k-means	50	-	-	23:54:43
	2-step	50	0.8	5	**01:40:46**
Dataset 1M	baseline	100	-	-	34:27:02
	k-means	100	-	-	Stopped after 72h
	2-step	100	0.8	5	**07:26:50**

Table 3.2 Performance on Dataset 500k and Dataset 1M, for the baseline model (SOM), k-means and the proposed 2-step approach. A maximum budget of 72 hours of computation has been provided.

k-means was stopped after 3 days, while the baseline SOM finishes in less than 35 hours and the 2-step approach in less than 8. This makes SOMs (and particularly with the proposed approach) an attractive alternative for very large datasets.

3.4 Discussion

As already argued at the beginning of the chapter, self-organizing maps have the great advantage of being linear in the input data size. Since it is a clustering algorithm, no labels are necessary, which implies that large quantities of data may be easily available (since no labelling overhead is introduced). Having a model that can handle large quantities of data – and can do so incrementally – can often be an important enough advantage to balance the lower performance that is typically obtained with SOMs w.r.t. other techniques. The work proposed in this chapter goes toward the direction of building a more efficient learning algorithm for handling unlabelled data: the proposed approach has slightly worse performance (in terms of accuracy and quantization error) when compared to a standard self-organizing map, but that comes at a significant reduction in training time – 2x to 8x, w.r.t. the standard (baseline)

approach. Larger datasets have been adopted to demonstrate how the 2-step training can make larger problems tractable.

We additionally argue that SOMs offer a more interpretable result w.r.t. other clustering algorithms, e.g. k-means. The interpretability of SOMs stems from the fact that points are projected to a 2D grid, where similar points are placed in an orderly manner. If the inputs are themselves easily understandable (e.g. images, such as MNIST digits), the SOM can be easily adopted to produce useful explanations.

The main limitation of self-organizing maps is due to the simple representations that can be built. However, there have been recent advancements in the topic, with the introduction of deep self-organizing maps [66] [67]. We find these approaches to be an interesting contribution worth investigating, to assess the extent to which optimized training approaches can be used for those models.

We conclude this chapter with a final consideration. It is often the case that machine learning problems are addressed by "throwing resources" at them: in other words, significant computational power is used instead of performing more calibrated operations. This leads to the adoption of many unsupervised algorithms that have a significant computational cost – at times even $O(n^2)$ and higher. SOMs already address this complexity by having a training cost linear in the number of data points available, and allowing to build an incremental model as new data becomes available. With the work discussed in this chapter, we would like to encourage a return to a more reasonable application of machine learning algorithms. Instead of applying algorithms "at all costs", we believe that each tool should have its own range of applications, and we show that – with some adaptations – the range of applications of many algorithms can be extended. Given the significant impact that machine learning can have on the environment (e.g. in terms of carbon footprint [68]), we hope that this direction will be, with time, shared more sistematically across the entire field.

Chapter 4

Explicit confidence-based pseudo-labelling

The core focus of this thesis it the study of the topic of learning from limited labelled data. In previous chapters we explored how the learning can occur with no labels at all, or with labels from adjacent domains. We instead focus this chapter on semi-supervised learning. The main assumption of semi-supervised learning is that there is a limited amount of labelled data available, but a large amount of unlabelled data can also be utilized for learning. This scenario is often encountered in practice, where the limited availability of labelled data is due to a shortage of human annotators, rather than a lack of raw data. Therefore, the goal of semi-supervised learning is to use the unlabelled data to improve the learning of the underlying data distribution, and then use the labelled data to iteratively propagate label information to new samples.

A recent approach called FixMatch [69] has been proposed for semi-supervised learning. FixMatch leverages both weak and strong data augmentations through consistency regularization, as well as the production of pseudo-labels for unsupervised samples that the model is confident in labelling correctly. This approach effectively combines the use of labelled and unlabelled data to improve the learning of the underlying data distribution.

FixMatch has demonstrated state-of-the-art performance on various datasets, including in situations where only a small number of labels are available. For instance, on the CIFAR10 dataset, FixMatch achieves 88.61% accuracy with only 40 labelled points, which amounts to 4 points per class.

In this chapter, we present a description of FixMatch and propose an alternative approach called ConFixMatch, which builds upon the strengths of FixMatch by further improving the confidence estimate used to produce pseudo-labels for the unlabelled data. The results presented in this chapter are preliminary and have not yet been published at the time of writing.

4.1 Related works

This work is an extension of FixMatch [69], which has been shown to achieve state-of-the-art performance on various semi-supervised learning tasks.

FixMatch exploits elements from both consistency regularization and pseudo-label assignment. Consistency regularization [70] [71] is a key element in recent state-of-the-art semi-supervised algorithms. The main idea behind it is that a robust model should produce similar outputs when provided with slightly perturbed versions of the same input. Pseudo-labels are assigned by a model based on what it has already learned. Because of this, pseudo-labels cannot be considered as having the same reliability as exact labels; since exact labels are typically human-assigned or anyway derived from a reliable source. Nonetheless, it has been shown that a semi-supervised model can benefit from being trained from both labels and pseudo-labels [72]. Pseudo-labels are typically used when the level of confidence that the model has in the prediction is high. In other words, if the model is unsure as to what the label should be for a given input, that input is not assigned a pseudo-label. The topic of model confidence will be explored in further detail in the rest of this work.

FixMatch is trained on a loss function that accounts for both a supervised and an unsupervised part. The supervised part of the loss function is a term that evaluates the quality of the prediction (e.g. in terms of cross entropy) of a weakly-perturbed input. These weak perturbations are applied to labelled inputs. Since the target class is known for that input, the cross-entropy can be computed easily. The second component of the loss function is an unsupervised one. Given an unlabelled input, the model first computes the predicted class for a weakly perturbed version of the input. Then, if the model prediction is "confident enough", the predicted label is used as a pseudo-label for computing the cross-entropy with the prediction over a *strongly*-perturbed version of the same input. Examples of strong perturbations are shown in Figure 4.1.

Fig. 4.1 Examples of weak and strong data augmentations on images. From left to right: the original input image (from CIFAR10), a weakly augmented version of the same image (simple operations are applied that preserve the main features of the image), a strongly augmented version (more significant operations are applied that affect the core contents of the image).

Thus, FixMatch exploits the confidence of the model in its predictions to improve the quality of the unsupervised part of the loss. The confidence of the model thus plays an essential role in understanding what should or should not be used for the training. Since a softmax activation function is applied to the output layer of FixMatch (much like it happen for virtually all neural network-based multi-class classifiers), the confidence used is the probability $p(y_i|x)$. In particular, the confidence c of a prediction for a point x is quantified as $max_i p(y_i|x)$. It follows that $c \in [0,1]$. A larger value corresponds to a larger confidence.

However, it has been shown in literature that neural networks are typically poorly calibrated in terms of confidence [73]. In particular, models are shown to be overly confident even for meaningless inputs [74], allowing for adversarial examples to be misclassified with high confidence [75]. This in turn affects the quality of what FixMatch decides to learn. In this thesis we explore an alternative way of computing the confidence of a classifier model and assess how that decision can affect the results achieved by FixMatch.

4.2 ConFixMatch

We thus present ConFixMatch (Confidence-based FixMatch), a semi-supervised learning algorithm which builds on top of FixMatch , adding a stronger estimation of the model's confidence to improve the overall performance. Instead of relying on the estimation of the confidence as the output of the softmax function, ConFixMatch produces a separate output, which explicitly states the confidence of the model itself in the prediction.

A similar approach is used in a different context in [76]. In that work, this novel definition of the confidence of a model is used to decide whether a data point is in- or out-of-distribution. We apply the same principle in ConFixMatch, as explained below.

Much like FixMatch, ConFixMatch receives an input and produces a probability distribution over all possible classes. For supervised samples, the input will be obtained as a weak perturbation of a point in the original dataset. For images, such perturbation may be rotations, flips, crops. For unsupervised samples, both a weakly augmented and a strongly augmented sample are produced from a single input. ConFixMatch produces, for the weakly perturbed input, both a prediction (probability distribution over all classes) and a confidence score.

If the confidence score provided by the model is large enough (i.e. if the model is confident enough in its prediction), the cross-entropy loss is computed between the strongly perturbed input and the pseudo-label predicted on the weakly augmented version of the same input. We note that the main difference with FixMatch is in the way in which the confidence score is computed: instead of being inferred from the output of the softmax, ConFixMatch now autonomously produces a separate output.

When the model makes a prediction having low confidence, ConFixMatch provides "aid" to the computation of the model's final output. We assume that $\hat{p}(x)$ is the original output of ConFixMatch, $c(x)$ is the estimated confidence and y is the ground truth class for sample x. We compute the final output of the model as:

$$p(x) = c(x)\hat{p}(x) + (1 - c(x))y \tag{4.1}$$

In this way, the aid is provided in the form of a "push" toward predicting the correct class: for the classification case, this amounts to increasing the predicted probability for the correct class, and lowering the predicted probabilities for all others.

ConFixMatch can thus now receive help in making the right predictions. Without the right incentive, ConFixMatch could collapse to a model always predicting an output with low confidence, thus receiving significant help and achieving good performance in terms of cross-entropy. We provide the incentive not to abuse the external help (i.e. to only use a low confidence when strictly necessary) by adding a penalty term on the prediction of large confidences.

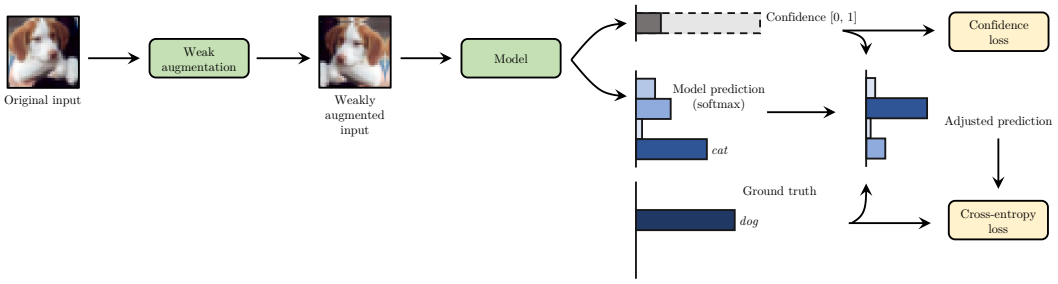


Fig. 4.2 ConFixMatch pipeline for an supervised sample. The weakly augmented sample is classified by the model. Differently from FixMatch, the model also produces a confidence score, which is used to adjust the final prediction.

Thus, if we refer to the original supervised FixMatch loss as ℓ_s, the loss for ConFixMatch will be $\ell_s - \lambda_{conf}\ log\ c(x)$. In this loss, the term λ_{conf} represents a penalty coefficient for using a low confidence. If λ_{conf} is too small, the model will receive close to no penalty when abusing the provided help, thus collapsing to a useless model. If too large, the model does not have any incentive in requesting help.

In practical terms, the confidence is implemented as a linear combination of the output of the second to last layer of the model, passed through a sigmoid function to produce an output in the range $[0,1]$. The work in [76] instead computes the confidence from a linear combination of the logits predicted by the model. We choose to instead use the output of an earlier layer to leverage a richer representation (i.e. a representation that has not already been reduced to unnormalized probabilities).

Figure 4.2 represents a diagram of ConFixMatch, when applied on a supervised sample. The confidence is produced separately from the class prediction. Two loss terms are extracted, to enforce the quality of the model and to produce meaningful confidences.

When the model is used with unsupervised inputs, it will still produce a confidence score for each point. In this case, no aid is provided when computing the cross-entropy: no ground truth is known for unsupervised inputs, making it impossible to provide any kind of help. The rest of the training occurs as with FixMatch: if the confidence is above a certain threshold, the corresponding strongly augmented sample is used for the unsupervised training (i.e. the cross-entropy is computed using as target the corresponding pseudo-label). Figure 4.3 represents a diagram for this situation.

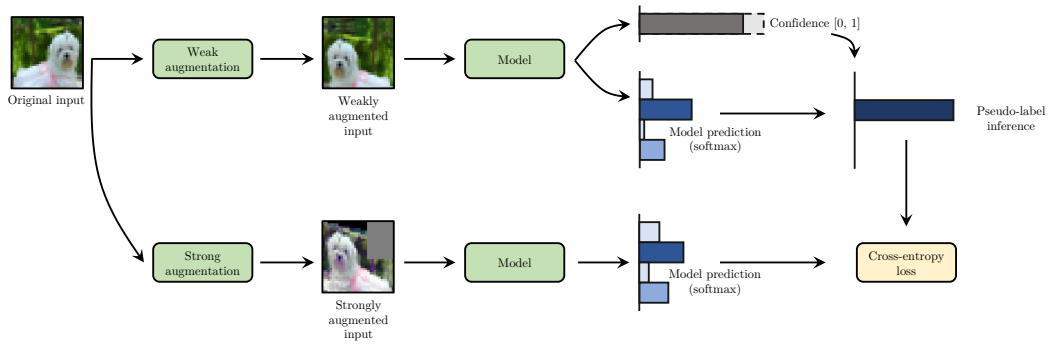


Fig. 4.3 ConFixMatch pipeline for an unsupervised sample. The weakly augmented sample is classified by the model. If the confidence predicted by the model is above a threshold, a pseudo-label is produced and used to train the model, with the strongly augmented sample as input.

Thus, the final loss function used is defined as follows:

$$\ell = \frac{1}{B}\sum_{b=1}^{B} H(p_b, p(y|\omega(x_b))) - \frac{\lambda_{conf}}{B}\sum_{b=1}^{B} log\ c(\omega(x_b)) + \\ \frac{\lambda_u}{\mu B}\sum_{b=1}^{\mu B} \mathbf{1}(c(\omega(x_b)) \geq \tau) H(arg\,max\ p(y|\omega(x_b)), \sigma(x_b)) \tag{4.2}$$

Where B is the batch size, μ represents the ratio between unsupervised and supervised samples in each batch. p_b represents the ground truth for point b, whereas $p(y|x_b)$ represents the model's prediction. $\omega(\cdot)$ and $\sigma(\cdot)$ represent respectively the weak and strong augmentation functions. $H(\cdot)$ is the per-sample loss function: in this study, the cross-entropy function is used, but it can be generalized to others, for different problems. τ is the threshold set on the confidence for an unsupervised sampled to be used for the training. Finally, λ_u and λ_{conf} are the two penalty terms for the unsupervised and the confidence portions of the loss, as already discussed. Their values can be set during the tuning of the hyperparameters.

We can expect the more significant benefits of using ConFixMatch to be obtained at early stages of training. The model can quickly learn to leverage a low confidence to receive the help needed to produce the final output. In this way, the predictions that will be made will have low confidence: a reasonable assumption given the early stages of training. Because of this, only few samples will be used for the unsupervised learning. By contrast, the confidence mechanism adopted by FixMatch

does not have an incentive to produce low confidence outputs at early stages, thus possibly introducing noise in the unsupervised learning process.

At later stages of the learning process we instead expect the confidence levels of the two approaches to stabilize. For longer training operation, thus, we do not expect to find significant differences between the two methods. However, given the importance of producing actionable models at early stages of training, we still find ConFixMatch to be a useful contribution.

4.3 Experimental results

In this section we present some preliminary results obtained at early training stages. In particular, we focus on a commonly used dataset in the field of image classification: CIFAR10 [64]. It contains 60,000 RGB images with a resolution of 32×32 pixels, belonging to one of 10 classes. The dataset is divided into a training set containing 50,000 images and a test set with 10,000 images. In this semi-supervised learning, we only adopt a fraction of training samples as supervised (labelled) ones, whereas the rest is used in an unsupervised manner.

Tables 4.1 and 4.2 show the top-1 and top-5 accuracies after 1,000 and 2,000 training iterations respectively, for various subsets of labelled data points (40, 250, 4,000). We mainly focus on early stages of training, where the benefit of an improved confidence approach is more evident. After 1k iterations, the benefits obtained with ConFixMatch are the most significant. The only situation where FixMatch appears to obtain better performance is in terms of top-1 accuracy, with 250 labelled samples. We note, however, that the 95% confidence intervals between the two methodologies overlap – thus making the difference less meaningful. For all other cases – and in particular with 4,000 labelled samples – ConFixMatch achieves significantly better accuracy performance.

We still observe this behavior after 2,000 training iterations. We note, however, that the difference in performance between the two methodologies reduces. This is in accordance with what was expected: as the two models improve in terms of performance, their confidence in the produced predictions will improve – thus both increasing the quantity of unsupervised samples used for training (i.e. above threshold) and their quality – with better pseudo-labels.

Method	40 labels		250 labels		4000 labels	
	Top-1	**top-5**	**top-1**	**top-5**	**top-1**	**top-5**
FixMatch	18.94 ± 1.16	67.21 ± 1.44	$\mathbf{33.75 \pm 1.58}^*$	84.70 ± 0.81	29.98 ± 1.78	84.40 ± 2.17
ConFixMatch	$\mathbf{23.51 \pm 1.06}$	$\mathbf{72.61 \pm 1.60}$	$31.79 \pm 1.69^*$	$\mathbf{87.02 \pm 0.69}$	$\mathbf{43.70 \pm 3.18}$	$\mathbf{92.11 \pm 1.56}$

Table 4.1 Comparison in terms of top-1 and top-5 accuracy between FixMatch and ConFixMatch, after 1,000 training iterations on CIFAR10. All results are computed on 5 separate experiments. In bold is the best performing algorithm. Starred results (*) show that the 95% confidence intervals overlap.

Method	40 labels		250 labels		4000 labels	
	top-1	**top-5**	**top-1**	**top-5**	**top-1**	**top-5**
FixMatch	$23.34 \pm 1.01^*$	69.59 ± 1.16	45.26 ± 0.82	90.53 ± 0.49	67.00 ± 0.95	$97.56 \pm 0.16^*$
ConFixMatch	$\mathbf{25.43 \pm 1.14}^*$	$\mathbf{73.64 \pm 1.87}$	$\mathbf{47.28 \pm 1.01}$	$\mathbf{92.12 \pm 0.34}$	$\mathbf{69.15 \pm 0.76}$	$\mathbf{97.71 \pm 0.27}^*$

Table 4.2 Comparison in terms of top-1 and top-5 accuracy between FixMatch and ConFixMatch, after 2,000 training iterations on CIFAR10. All results are computed on 5 separate experiments. In bold is the best performing algorithm. Starred results (*) show that the 95% confidence intervals overlap.

4.3.1 Confidence assessment

The main focus of this part of the thesis is on improving an existing semi-supervised methodology by changing the confidence mechanism used. We have shown that the first empirical results appear to be promising, with a significant improvement in performance at early training stages.

We additionally study the behavior of the confidence estimation at early training stages, i.e. when the model first learns how to estimate the confidence. These are the crucial moments where a better confidence estimation can better bootstrap the learning process.

The main driver of the confidence is given by λ_{conf}, the regularization parameter that prevents the model from abusing the confidence as a way to obtain low cross-entropy scores. As already discussed, a larger λ_{conf} will result in a model that is penalized more when requesting "help". We study how the model estimates the confidence throughout the first learning stages.

In particular, Figure 4.4 shows the average confidence for the first 1,000 training batches, for various values of λ_{conf}. As expected, a larger penalty term results in a model that starts producing larger confidences, so as to avoid an excessive penalty. For comparison, the confidence used by FixMatch is reported as well. Since the threshold τ can be adjusted, a comparison in terms of values obtained by the various

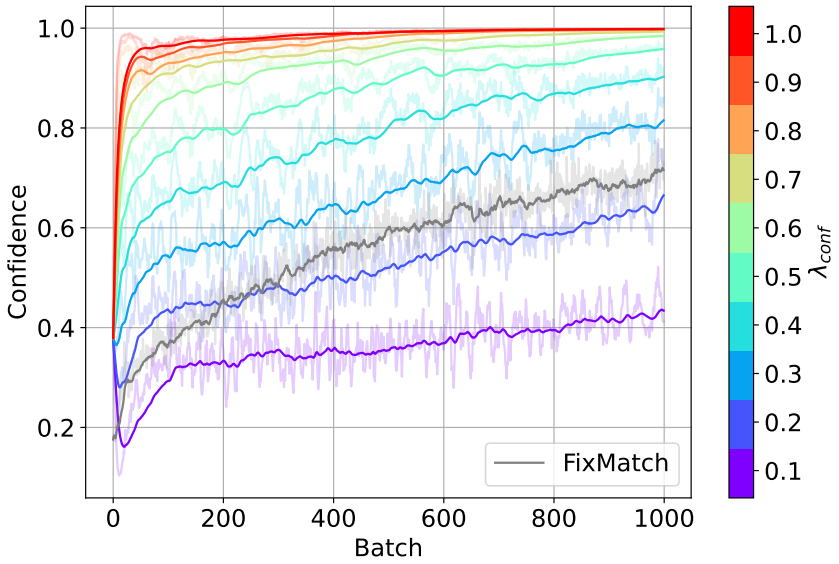


Fig. 4.4 Evolution of the average confidence score for each batch, as λ_{conf} changes, for a model trained with 4,000 labelled samples. In transparency are the actual mean values (confidence intervals not reported for graphical reasons). The overlaid line represents the exponential moving average of the mean values. In gray is the confidence for FixMatch's default configuration.

algorithms is not particularly meaningful. What is interesting is instead the fact that FixMatch has a slower transitory, whereas ConFixMatch quickly converges to a stable confidence.

While this plot provides an interesting insight in terms of behavior of the confidence throughout training, it does not help with the identification of the best-performing value for λ_{conf}. We note that, as already mentioned, the effect of λ_{conf} on the model is also guided by the choice of threshold τ. Figure 4.4 shows that the confidence converges to various asymptotic values based on λ_{conf}. Although it could be reasonable to tune both τ and λ_{conf}, we instead decide to fix τ and find the λ_{conf} for which the best performance is achieved. Through a preliminary tuning phase we identify the best-performing model (in terms of top-1 accuracy on a validation set) for a value of $\lambda_{conf} = 0.6$. The results in Tables 4.1 and 4.2 have been obtained with this choice.

4.4 Discussion

In this chapter we presented preliminary results for ConFixMatch, a novel version of the popular FixMatch semi-supervised approach. These initial results show a promising new direction that appears to be worth investigating. By introducing an explicit prediction of the confidence level of the classification made, ConFixMatch is able to better choose the unsupervised samples for which the pseudo-label can be used.

This chapter only covers initial results, additional experiments will be needed to be able to assess the extent to which a novel definition of the confidence can affect the performance of semi-supervised techniques. These initial results also seem to indicate that one of the problems of the original confidence is in its slow convergence. Because of this, it may also be interesting to study several variations of the original FixMatch algorithm with various confidence convergence properties enforced. We additionally note that other policies to guarantee a reasonable confidence may be enforced: in this work, we focused on a reward in terms of cross-entropy. We may additionally consider reducing the penalty on the cross-entropy for low-confidence predictions (e.g. by weighting the cross-entropy with the predicted confidence). In this way, confidently incorrect predictions will significantly affect the overall loss, whereas incorrect predictions with low confidence will have a reduced penalty.

The results presented in this work only focus on one dataset (CIFAR10) and only at early training stages. The natural extension is to focus the future research to exploring the extent to which the proposed confidence approach generalizes to other datasets, and the impact that it has on training time. The initial hypothesis that we advance in this thesis, and that will require additional testing, is that this novel approach to confidence is beneficial especially at early training steps. We can expect the original methodology to catch up with ConFixMatch at a later time, when the model produces more reliable (i.e. high confidence) pseudo-labels. Nonetheless, the goal of producing models that reach convergence faster is an important one, both from a learning perspective (all else being equal, the model that reaches convergence faster is the one with the most efficient learning process) and from an environmental one (shorter training implies a lower carbon footprint [77], as already argued in Chapter 3).

Chapter 5

Limited labelled data: a case study

In this chapter, we present a case study where data is scarcely available and, for what little is available, it is unclear how it should be labelled. This is far from the well-defined tasks that are typically identified within an academic setting. However, it is a rather recurring occurrence in industry.

In particular, we will discuss a predictive maintenance (or prognostics) scenario, carried out in collaboration with General Motors. In this predictive maintenance task, we need to identify situations where a sensor drifts from its nominal behavior, and notify this change before this malfunctioning causes undesirable behaviors. The task, however, is problematic for different reasons. First, the goal of the prediction is ill-defined: as we will see, it is unclear how the problem should be framed and, even so, how to obtain meaningful supervision (as none is explicitly provided). Second, the amount of data available is limited, both in cardinality and in variability.

As it will be shown, we propose an approach to build pseudo-labels for the target problem (low, medium, high risk) based on externally available data and a semi-supervised process. In this way, we can apply commonly adopted classification techniques without the additional overhead of requiring human-assigned labels for all points in the dataset.

In the introduction of this chapter we provide the context surrounding the problem. Next, we explore how the problem is framed and how it might be approached by addressing all aspects from data acquisition to model evaluation.

With the Internet of Things paradigm, data collected from vehicles (e.g. cars, trucks) can be collected locally and sent to a remote storage location for later analyses with tools other than the limited computing available on-board. Hence, automotive manufacturers can leverage the data collected by their own on-board systems to offer additional value to their customers. This value can be defined in terms of additional services and features – thus improving the convenience for the customer.

One of the additional services that can be offered is predictive maintenance. Predictive maintenance aims at the identification of possible malfunctions ahead of time, allowing a prompt intervention before the actual failure. Both manufacturers and customers can benefit from this kind of prediction. The former can issue recalls only when actually needed and before irreversible damage occurs, the latter will not experience unexpected vehicle malfunctions. For these reasons, automotive companies are actively interested in predictive maintenance.

General Motors (GM) has been a leader in the application of automotive prognostics, which is marketed in the US since 2015 under the name of OnStar Proactive Alerts. The proactive alerts presently cover the vehicle starting system on millions of production vehicles. However, these alerts all follow the model-based (or physics based) approach, whereas this work is focused entirely on a data-driven one.

In this work of thesis we focus on studying predictive maintenance solutions for the oxygen sensor (also known as lambda sensor). This sensor, placed on the exhaust system of combustion engines, measures the fraction of oxygen in the output gases. This information allows the ECU (Engine Control Unit) to optimally regulate the ratio of fuel and combustion air for the catalyst, reduce the emission of pollutants, and optimize the injection system's performance.

Due to the imperfect burning of the combustion, the engine ejects some soot, which accumulates and clogs the oxygen sensor. As a result of clogging, slower and incorrect oxygen measurements cause a sub-optimal performance of the injection system and increase harmful emissions. Some engine operations can clean the oxygen sensor. These could be triggered periodically or when a pre-alarm status is identified. If not correctly handled, the oxygen sensor gets too clogged, and the ECU turns the check engine light on, forcing the driver to go to the service for costly maintenance operations.

Hence, the early prediction of the pre-alarm status of the oxygen sensor clogging is fundamental to trigger the cleaning operations. Unfortunately, measuring or

predicting the oxygen sensor status is a complex task due to the many factors that drive the soot accumulation process, including driving style, engine age, vehicle load, fuel quality, weather conditions, etc.

In this work of thesis we describe PREPIPE a data-driven framework that, given a large number of time series collected from a vehicle's ECU, builds a model to predict if the sensor is currently unclogged, almost clogged (since the clogging of the sensor happens gradually), or clogged.

The aspects of interest of this work from an industrial perspective have already been argued. From a research standpoint, the main contributions to this thesis are:

- *Building and end-to-end pipeline*, that starts from raw data and produces actionable results, and that is validated in multiple ways to guarantee the quality of the results obtained,
- *Producing a semi-supervised labelling technique*, which leverages a hybrid domain- and data-driven approach to produce useful labels that relieves the human from taking part in an intensive labelling process,
- *Proposing an unsupervised signal selection algorithm*, that can be applied to other contexts (supervised or not) and that produces comparable performance with other fully supervised approaches.

The work presented in this chapter also offers additional contributions of interest. While not specifically relevant to the topics of this thesis, we still believe them to be particularly useful and will be presented, for the sake of completeness.

The major contribution reported in this thesis have been published in [16] and [17]. Additionally, a patent has been filed by General Motors regarding the main aspects of interest of the proposed approach [78].

5.1 Related works

This section outlines related predictive maintenance works in the automotive domain, highlighting common challenges of the field and the proposed solutions. We additionally analyze works carried out using deep learning approaches, which have

gained significant momentum in recent years. Finally, we analyze specific studies facing the oxygen sensor diagnosis.

As already discussed, in this work a main aspect of interest is the transformation of the problem at hand into a supervised one. We do so by assigning pseudo-labels to the available data. For thoroughness, we explore both unsupervised and supervised techniques that have been proposed in the literature for tackling predictive maintenance tasks.

Predictive Maintenance in the automotive field. The increasing capability to collect vehicles data has fostered many studies to monitor, detect, and define predictive maintenance operations in the automotive industry. The authors of [79] present a complete overview of prognostics and health management in transportation and the automotive industry.

Due to the ECU's limited memory, modest computational capability, and bandwidth constraints, a common trend is that of leveraging data dimensionality reduction techniques. Similarly to our proposed pipeline, some works focus on the feature selection step for machine learning techniques using approaches such as multiway partial least squares [80], common factor analysis [81], a combination of domain expertise and PCA [82], wrapper feature selection, and filter methods based on the Kolmogorov-Smirnov test [83], or minimum redundancy maximum relevance algorithms [84]. Differently from these works, in this thesis we tackle the problem of feature engineering by integrating and evaluating a wide range of signal selection, feature extraction, and feature selection approaches, thus producing a general solution that can be applied to different scenarios as needed. Compared to previous works, we aim to offer interpretable results to the domain experts, who supervised the entire process and gained useful insights on the phenomenon under study.

To complete the predictive maintenance pipeline, we rely on well-established machine learning algorithms commonly used in the predictive maintenance context [85].

Deep learning-based predictive maintenance. Recently, deep learning models have been getting popular for fault diagnosis and prognostics. Such shift is driven by the generally superior predictive performance and their capability of handling high-dimensional data in predictive maintenance and health management scenarios [86–88]. These approaches directly work on the raw input data with very little need for any feature engineering.

Restricting to automotive applications, the authors of [89] propose a data-driven deep learning diagnostic approach based on a combination of convolutional (CNN) and long short-term memory (LSTM) neural networks. They use ECU data to detect pre-ignition, i.e., ignitions before the spark plug fires. The authors in [90], instead, introduce a combination of the dual-tree complex wavelet transform, coupled again with CNN and LSTM, to monitor the health status of a vehicle suspension. Given the need for domain experts to understand both the prediction process and the phenomenon characteristics, we prefer to follow a well-designed predictive maintenance pipeline. We show it offers performance comparable with state-of-art deep learning architectures while satisfying the interpretability requirements.

In terms of unsupervised learning, autoencoders are a commonly adopted technique for anomaly detection in predictive maintenance [91]. Autoencoders are used to build dense representations of a piece of data, i.e. with a lower dimensionality w.r.t. the original one: for example, the signals read from an engine are projected to a low-dimensional space that contains the most salient features of such signals. The original data is then reconstructed. Outliers are identified as those situations where the reconstruction error is particularly large (e.g. above a given threshold): this occurs because the piece of data to be reconstructed is typically sampled from a different distribution than the "nominal" data – and as such should be considered as an outlier.

While autoencoders have produced meaningful results in completely unsupervised scenarios, we observe that even weakly supervised techniques typically produce much better results for outlier detection tasks, as shown in [92].

Oxygen sensor diagnosis. The oxygen sensor plays a vital role in reducing exhaust emissions, and it is crucial to monitor and diagnose its status and detect and predict faults. To this extent, different works focus on the predictive maintenance of such key elements. For instance, authors of [93] propose a model-based solution and use the exhaust pressure pulsation to detect deterioration of oxygen sensor dynamics caused by sensor clogging. Similarly, the authors of [94] leverage data collected by a UEGO (universal exhaust-gas oxygen) sensor in a controlled environment, apply a PCA to identify the most relevant signals and a feed-forward neural network to detect if the oxygen sensor is faulty.

We instead propose a full pipeline that covers all steps from data acquisition to model predictions in significant more depth.

5.2 The oxygen sensor case study

The oxygen sensor is a device used to measure the proportion of oxygen in the exhaust gases of an internal combustion engine. This information is fundamental to lower the exhaust gas pollutants and optimize the performance of the injectors' fueling system and engine in general.

Due to the accumulation of the unburnt hydrocarbon-based soot contained in the exhaust gases, the oxygen sensor is subject to clogging. When the sensor is clogged, slower and incorrect oxygen measurements cause a sub-optimal combustion efficiency resulting in more harmful emissions released into the environment. Currently, the ECU can only diagnose when very slow oxygen measurements occur, i.e., when the oxygen sensor has reached a critical state, and its readings are unreliable. When this situation occurs, the ECU turns the check engine light on, and the driver has to go to the service for the required maintenance operations, i.e., a costly manual sensor cleaning operation.

While the clogging process is a known problem, its non-linear and non-monotone trend makes its prediction a complex task. Indeed, while it is well known that the clogging increases slowly over time due to the soot accumulation, driving conditions, fuel quality, and driving styles affect this process. For instance, sudden abrupt accelerations, vibrations, or specific engine operations like active regeneration [95] can partially clean the sensor by burning or detaching some soot from the oxygen sensor. Some of these operations can be triggered by the ECU to clean the sensor before it reaches a critical status. As such, an early prediction of the oxygen sensor's clogging status is fundamental to run these specific engine operations and clean the sensor to avoid malfunctioning.

5.2.1 Dataset

We adopt data collected in a test bench where an actual diesel engine equipped with the standard on-board and some additional external sensors. The testbed lets us simulate real driving conditions with different vehicle loads and conditions. In detail, in each experiment, called *cycle*, the simulator follows a "driving pattern", i.e., a predefined sequence of gas pedal presses and releases coupled with different engine loads to reproduce different driving situations (e.g., urban, extra-urban, highway).

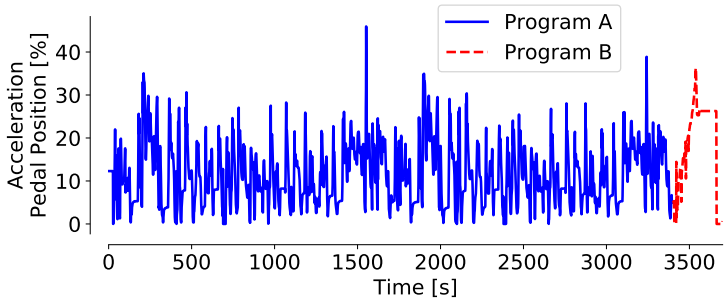


Fig. 5.1 Acceleration pedal signal, recorded with both Program A (in blue) and Program B (in red).

For our experiments, we follow the driving cycle derived from the Real Driving Emissions (RDE) test procedures [96]. We focus on the RDE procedures as, currently, they are used to create the standard homologation cycles for testing particles and exhaust emissions in real traffic and environmental conditions. They are designed to represent real driving situations as closely as possible.

We employ two different data loggers to monitor the engine under test, namely Program A and Program B (names redacted under request of GM). These programs have different characteristics, and we leverage them for similar – yet complementary – purposes. Program A monitors the engine during the entire cycle as in a real on-board scenario. It records a set X of 50 signals related to on-board and bench sensors. Each signal x_i is a time series where samples $x_i(t)$ are collected with a frequency of 1 Hz.

Program B, instead, monitors only the final 5-minute period of each cycle when a specific input sequence is imposed (described later) and provides more details about the engine and oxygen sensor behavior by collecting hundreds of bench signals at a higher frequency (320 Hz). As we will discuss, the data collected from Program B (which is not available for deployed vehicles because of the nature of the sensors used) will be leveraged to apply the pseudo-labels.

Figure 5.1 depicts the trace of the gas pedal position during the entire cycle highlighting in red the part monitored by both programs. Each cycle last 62.5 minutes (3750 seconds). In total we obtain a dataset D with 388 cycles. Table 5.1 reports the main characteristics of these programs.

As already briefly mentioned, the data from Program B can only be collected in a test bench, due to two reasons: the high sampling frequency used and the fact

	Program A	**Program B**
Duration	3750 s	300 s
Sampling frequency	1 Hz	320 Hz
Number of signals	50	440
Number of cycles	388	388

Table 5.1 Program A and Program B characteristics.

that Program B makes use of external sensors added for the data collection phase. We thus only use the data from Program B to assign pseudo-labels to the cycles, as discussed in Section 5.4. The rest of the work is instead focused on working with the data collected from Program A, since the same data can also be collected from real vehicles.

Only a subset of the 50 signals recorded by Program A are useful to predict the clogging status of the oxygen sensor. As such, we perform a preliminary data selection procedure to remove all the test bench signals that are not available on real vehicles. Then we discard signals unrelated to the problem with the support of the domain experts that properly consider the informativeness of the signals. Next, we discard signals with constant values (e.g., alarms) that carry no information. Finally, we keep only one among possible sets of strongly correlated signals (i.e. having a Pearson correlation of ± 1).

For part of this study, we additionally remove the signal obtained from the oxygen sensor, since we are interested in identifying other behaviors within the engine that correlate with the oxygen sensor behavior. We will re-introduce the oxygen signal to draw some useful conclusions in Section 5.9.

As a result of this a-priori data selection procedure, we obtain a set $\hat{X} \subset X$ of 30 signals.

5.3 Predictive maintenance pipeline

We designed the PREPIPE early PREdiction PIPEline for the oxygen sensor clogging represented in Figure 5.2. In this section we summarize each step of the PREPIPE pipeline, while a detailed description is provided in Sections 5.4 and 5.5.

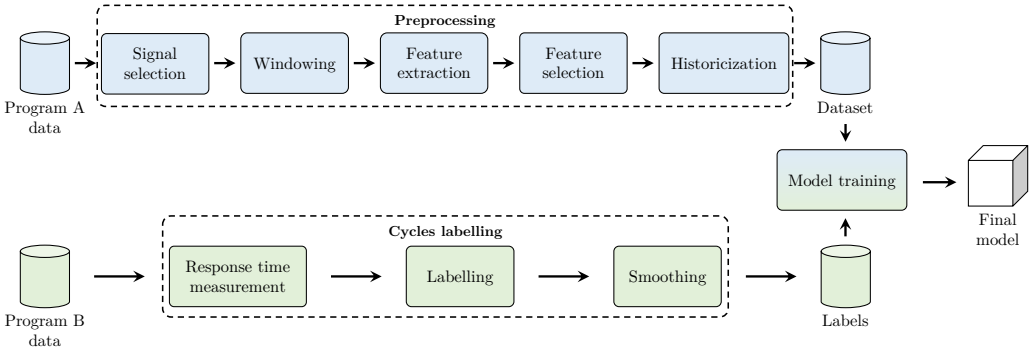


Fig. 5.2 The PREPIPE predictive maintenance framework. The top pipeline represents a classic supervised problem. The bottom pipeline represents the semi-automated label extraction process. The final model training combines the results from the two pipelines.

Signal selection. The input to the pipeline is the set of signals recorded in each engine cycle by the on-board sensors. Given the large number of available signals, we evaluate different supervised and unsupervised learning algorithms to select the best subset of signals describing the oxygen sensor status.

Windowing. We determine the correct size of the time window to be used for the monitored signals. The window size should allow an accurate evaluation of the oxygen sensor status.

Feature extraction. Signals are recorded as time series representing the variable values during the time window. Signals are transformed into *features* by means of different feature extraction strategies. These features allow us to represent characteristics of the time series that would not be visible in a "sample-based" representation.

Feature selection. Several features represent each signal, some of them possibly redundant or uncorrelated with the target variable. We reduce the number of these features through a supervised feature selection stage.

Historicization. Each time window describes the current status of the oxygen sensor, and it does not include any historical information about previous windows. Hence, we add historical features related to the past cycles evaluating the benefit (if any) of including past observations.

Response time measurement. Since labels are not directly available for the cycles, we use some of the properties of the cycles, as measured with Program B, in addition to some domain expertise, to extract a response time from the signals. This quantity will later be used for the extraction of a class label.

Labelling. At this step, the response time (a continuous variable) is discretized into one of three classes. The discretization process is automated and based on the identification of thresholds, through the application of a 1-dimensional k-means clustering approach. The thresholds are later validated by the domain experts.

Smoothing. The automated extraction of class (pseudo-)labels is necessarily a noisy operation (if it was not, the problem would have been trivial). We introduce an assumption of continuity in the observed phenomenon. This allows us to smooth the behavior of the response time, thus producing more consistent class labels.

Model training To model the clogging status we adopt an artificial neural network model [97]. Given the cumulative nature of the clogging phenomenon, we use two validation strategies to determine whether the temporal order plays a crucial role in the prediction of the oxygen sensor status or each sensor status decision is independent.

At each step, we select the best option and the parameter setting through a *wrapping approach* [98] in which a classifier is used to identify the best choice by comparing the predictive performance with the variation of the step configurations. In short, we run the complete pipeline from the signal selection to the model training, tuning and validation, by sequentially optimizing one step at a time. Initially, we assign *default values* at each step to identify a baseline of the performance. Then, we independently optimize the parameters following the natural sequence of the steps in the proposed pipeline.

5.4 Cycles labelling

The available cycles have been collected by General Motorsfor a purpose other than predicting the oxygen sensor status. The data collection phase is a time-consuming and expensive operation: it involves allocating a testing room for possibly several weeks, with an engine running 24/7 and dedicated personnel. For this reason, data may be collected for one goal and repurposed afterwards, as is the case with this work. This project stems from an attempt to use that same data for a different approach.

In other scenarios, an experiment may be devised specifically for the collection of the data of interest. In that case, the “label” (i.e. the engine status) may be enforced

by the designers of the experiment. It is, for example, the case in [84], where we approached another predictive maintenance task with GM: in this case, the data collection phase was explicitly designed to address a High-Pressure Fuel System failure, by systematically inducing any desired engine condition.

Since the available data was collected for a different purpose, there is no clear label that can help us quantify the degree to which the oxygen sensor is clogged.

However, with an intuition that comes from the domain experts, we can devise a labelling procedure that can help us automatically label points (cycles). In particular, we know that when the oxygen sensor is clogged, the readings of the sensor will be slower, all else being equal. We additionally know that all cycles follow the same "track" (i.e. the same sequence of operations on the gas pedal). This reduces the variability of the collected data – which may problematic when trying to generalize to unseen data – but it provides a great advantage: we can identify a sequence of operations that will be executed in the same way across all cycles. If this sequence of operations can be found in the final part of the cycle (i.e. the one also measured with Program B, at a higher sampling frequency), we can accurately measure the variation in measured oxygen concentration and infer which cycles have a slower response. We take this intuition and further elaborate it in the following subsections. More specifically, there are three main steps to be taken for this semi-supervised labelling process:

- *Response time measurement*: the target sequence of operations is identified and we define a standard approach to quantifying the responsiveness of the oxygen sensor. We call this quantity the "response time".
- *Labelling*: we approach this predictive maintenance problem from a classification perspective, as such, we define some classes of behaviors of interest and discretize the response time accordingly.
- *Smoothing*: due to the semi-supervised nature of the process, the labels obtained contain some noise. By making some additional assumptions on the nature of the problem under study, we try to reduce the resulting noise through a smoothing procedure.

Response time measurement

The oxygen in the exhaust gases (i.e. the oxygen measured by the sensor) is directly influenced by the state of the gas pedal, since it drives the load on the engine: when the combustion occurs at a faster rate, more oxygen will be burned, thus resulting in less oxygen being found in the exhaust gases.

Because of this, if we identify a situation where a significant change in the state of the gas pedal occurs during the final portion of the cycle (because of the higher sampling rate of Program B), we can quantify the rate of change of the oxygen measurement. Through visual inspection, we can identify one such pattern, where the gas pedal is first kept at a steady level for a prolonged period of time, only to be completely released. This operation, which will be referred to as a "cut-off", is shown in Figure 5.3.

The oxygen level passes from an initial oxygen level, $O_{2_{start}}$, which depends on the gas pedal position (close to 9% for the adopted track), to a final value $O_{2_{end}} \approx 21\%$ (close to the oxygen concentration in the atmosphere). This transition occurs for all cycles in the same way, since it is a response to the same step input applied to the gas pedal. Because of this, the only variable that can affect the *measured* duration of the transitory is the extent to which the oxygen sensor is clogged: a longer measured duration implies a more clogged sensor.

The time constant of a system is defined as the time necessary for a system to reach $\approx 63\%$ $(1-1/e)$ of the final value, in response to a step input. We use this definition to identify the *response time* of the engine, for each cycle, as shown in Equation 5.1. Figure 5.3 provides a visual representation of the response time.

$$t_r = t(O_2 = \left(1 - \frac{1}{e}\right) \cdot (O_{2_{end}} - O_{2_{start}})) - t(O_2 = O_{2_{start}}) \tag{5.1}$$

The trend of the response times measured on Program B across cycles is shown in Figure 5.4. It is immediately clear that the response time varies in a range of values between 1 and 2 seconds. As such, measuring the response time with Program A (at a sampling rate of 1 Hz) would not have provided a meaningful result. For this reason, the data from Program B is particularly useful, even if only available in the test bench.

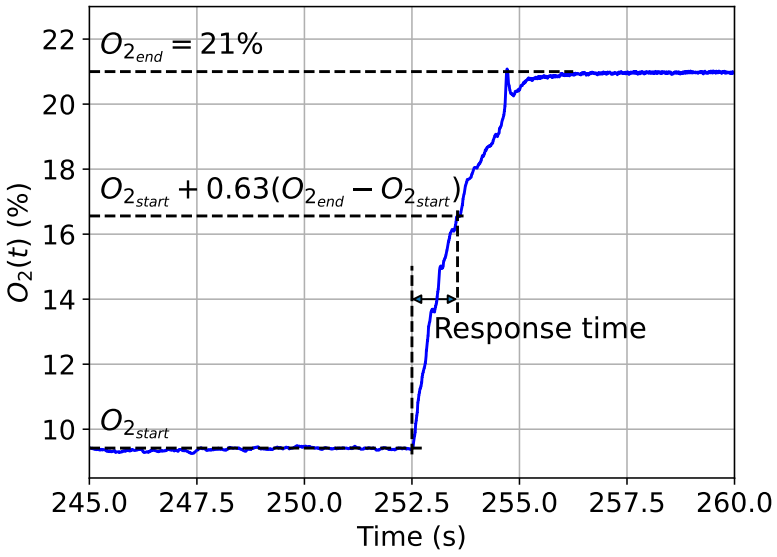


Fig. 5.3 Response time measurement process

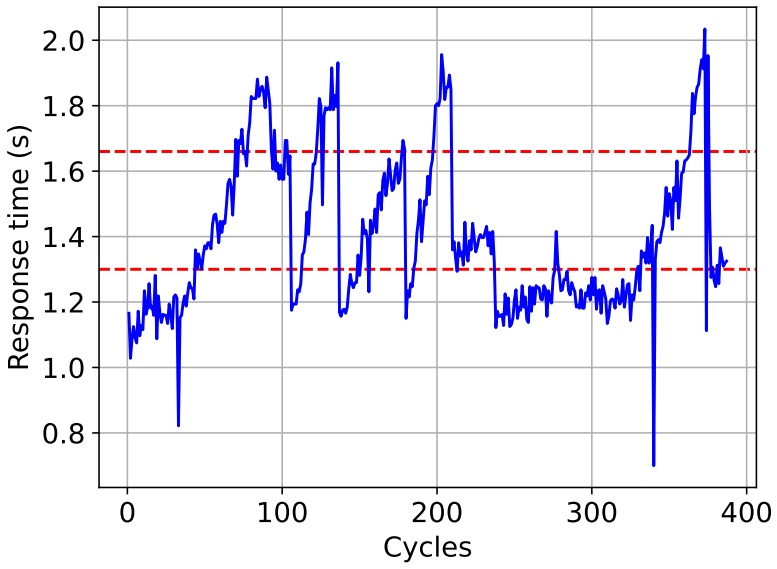


Fig. 5.4 Response time trend throughout the experiment. The horizontal lines show the positions of the thresholds

Labelling

The response time represents a pseudo-target we can work with in a supervised manner. However, the final output of the model should be discrete, i.e. the level of "alert" that should be issued. We decided to identify three levels of warning: *green* for a nominal behavior, *yellow* for a behavior that has drifted from nominal but not problematic, and *red* for situations that significantly differ from the expected behavior. To preserve the pipeline automation, we use an unsupervised approach to identify these three classes. More specifically, we apply 1-dimensional k-means clustering [99] to produce three classes.

For 1D k-means, cluster centroids can be ordered: assuming that centroids c_{green}, c_{yellow}, c_{red} are identified, we can alternatively define 2 thresholds, $(c_{green} + c_{yellow})/2$ and $(c_{yellow} + c_{red})/2$, which can be similarly used to define the bins for the discretization of the response time. These thresholds are shown as horizontal red lines in Figure 5.4. Figure 5.5 shows the distribution of response times based on these newly defined thresholds. The only human interaction in this process comes at the end of this labelling process, where the domain experts manually validate the quality of the identified thresholds, and can optionally introduce minor changes to them if deemed necessary. This is in contrast with the significant effort that would be required to either (i) design and execute a data collection process aimed at collecting the relevant data, or (ii) manually label each of the already available 388 cycles.

Smoothing

As already discussed, Figure 5.4 illustrates the response time evolution throughout the cycles of the experiment. Due to the automatic label extraction process, we note that the response times observed have a noisy component – as highlighted by the spiky profile of the curve. We operate under the assumption that the soot cumulation process that results in the clogging of the sensor has a behavior that is continuous in nature.

Based on this, we introduce an additional step to help improve the quality of the pseudo-labels assigned: a low-pass filter in the form of a moving average is applied to each response time in the sequence. We study how this affects the assigned labels

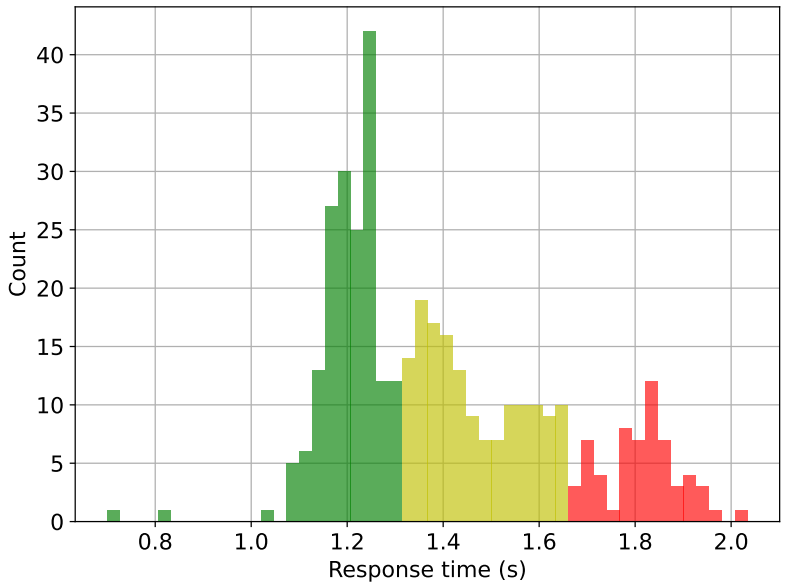


Fig. 5.5 Distribution of response times: the colors represent the assigned classes based on the defined thresholds

as the window size changes. We consider averaging each point with the previous w and the following w values, for a window size of $2w+1$.

The desired behavior is that only a limited number of "label switches" occur close to the threshold values. Higher values of w render the response times curve smoother, thus reducing the total number of switches, as shown in Figure 5.6. The knee of the curve indicates an acceptable value.

Based on this curve, we notice that a value for $w = 2$ may be reasonable. We use a visual inspection of the profile of the smoothed curve with this value to understand whether the resulting alteration in values is acceptable or not. Figure 5.7 shows the results for 4 different values of w (0 through 3): the vertical bands are colored based on the assigned label for each cycle. In general, the profile of the curve for $w = 2$ has been deemed acceptable from this perspective as well.

From the smoothed values, the labels can be assigned to each Program B cycle: the cardinalities for the three classes are reported in Table 5.2. From this table, we observe that the red class, which is the one of most interest, is also the minority one. This is to be expected, given that anomalous behaviors are typically a minority w.r.t. nominal ones. This will require additional considerations when building the rest of the pipeline. Although the pseudo-label is inferred on the final portion of each cycle,

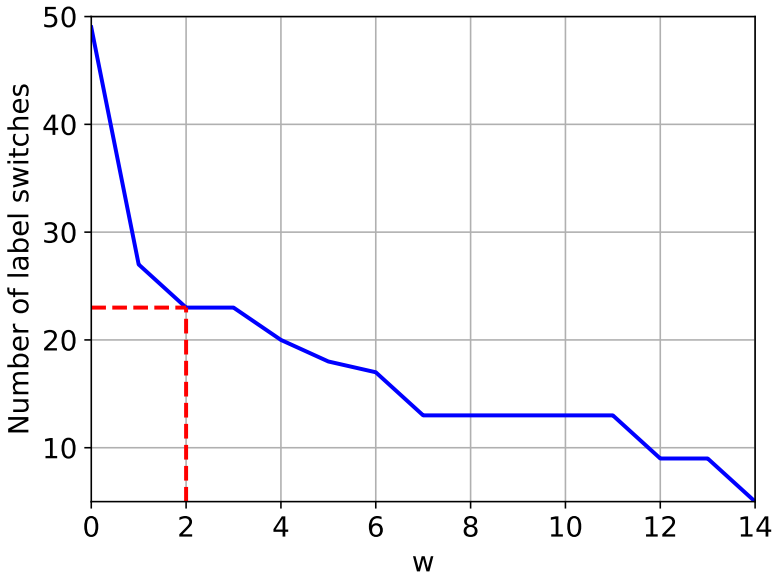
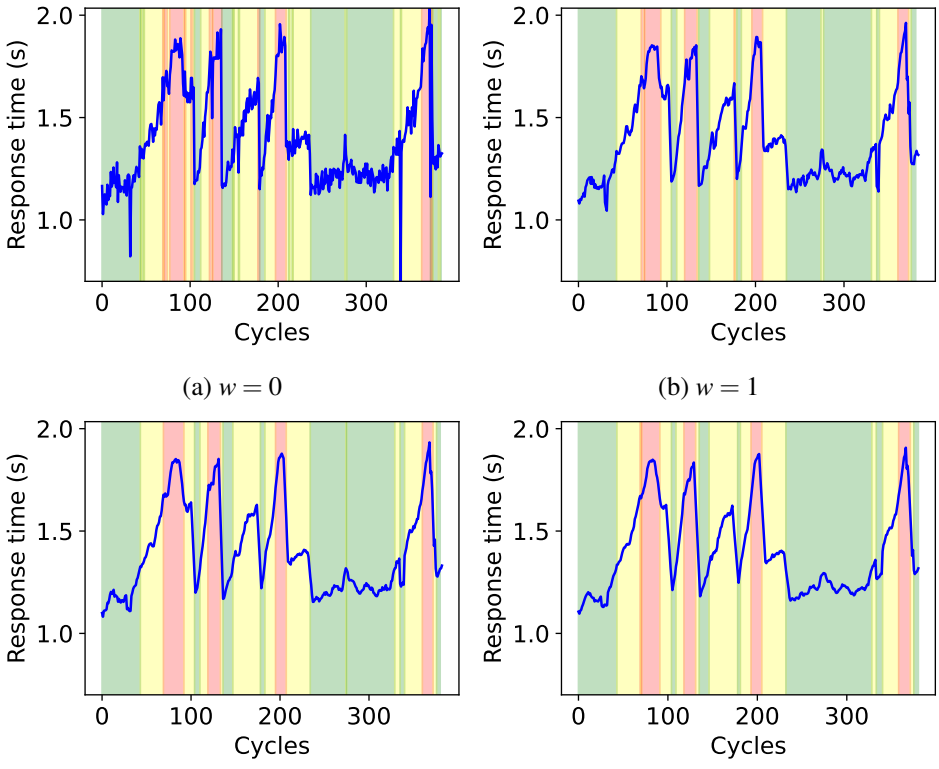


Fig. 5.6 Number of label switches occurring as w increases



(a) $w = 0$ (b) $w = 1$

(c) $w = 2$ (d) $w = 3$

Fig. 5.7 Response time trend smoothed with different w values

Class	Cardinality
Green	164
Yellow	163
Red	61

Table 5.2 Cardinalities for the three identified classes

it is reasonable to assume (given the slowly changing nature of the phenomenon) that the label can be propagated to the entire cycle (i.e. the engine is assumed to be having the same condition throughout the entire duration of the cycle – approximately 1 hour).

With the proposed semi-supervised labelling approach described, we are able to introduce supervision through the generation of pseudo-labels. These pseudo-labels are obtained from a process that only has weak human supervision and effectively allows us to repurpose data collected for a different objective. For the rest of this chapter, we will typically refer to the pseudo-labels obtained in this way as labels, with the implied meaning that they are not human-assigned and so subject to possible noise. The rest of the chapter presents the rest of the proposed pipeline, that is supervised in nature for most aspects.

5.5 Preprocessing

The initial steps of the processing pipeline address all the tasks required to prepare the data of a given cycle for the model training, tuning and validation steps.

Signal selection

The first step consists of selecting the best subset of signals to feed the classifier. This brings several advantages: (i) *improved data collection on the field* by reducing the costs required for the on-board hardware and the bandwidth needed for the data transmission to a centralized server, and (ii) a *more concise representation of each cycle*, which lowers the dimensionality of the problem, thus simplifying the rest of the pipeline.

For the signal selection step, we aim at discarding entire signals that are not deemed useful. For this, we test different unsupervised and supervised learning algorithms. These algorithms either produce a ranked list of signals, from the most to the least important one, or already produce a curated list of signals to be preserved, and a list of signals to be discarded. In the end, starting from the set $\hat{X}$ we aim to get the best possible subset $\bar{X} \subset \hat{X}$.

In addition to testing various existing supervised and unsupervised technique, we propose CORR-FS (Correlation-based Feature Selection), an unsupervised algorithm that iteratively identifies the most representative signals among a set, and discards redundant ones.

To discover redundant signals CORR-FS analyses the correlation among them. Given any two signals, the Pearson correlation coefficient between the two sequences can be used as an indication of how related the two variables are. For each pair of signals (i,j), CORR-FS computes the correlation coefficient through the Pearson correlation defined as $\rho_{i,j} = \frac{\text{cov}(i,j)}{\sigma_i \sigma_j}$, where $\text{cov}(i,j)$ is the covariance between i and j, σ_i is the standard deviation of i and analogously σ_j for j. Figure 5.8 shows the correlation matrix plotted as a heatmap of the correlation coefficients computed for each pair of variables for one of the available cycles. At a glance, the plot shows that significant redundancy exists within the available signals.

To properly handle the data redundancy and iteratively extract the most representative signals from the initial pool of available ones, CORR-FS identifies a list of independent signals (where "independent" can be defined in terms of correlation) ordered by descending degree of representativeness of the other signals. Specifically, CORR-FS requires a single parameter r_{min} to be selected. This parameter represents the minimum correlation coefficient below which two signals are considered as not strongly correlated. The value for the r_{min} parameter has been defined empirically (see Subsection 5.7 for more details). CORR-FS performs the following steps:

1. For each pair of signals i and j, their overall correlation coefficient $r_{ij} = r_{ji}$ is computed as correlation coefficient between the samples collected from the two signals, across all cycles.

2. The set of "remaining signals" $L^{(t)}$, which will be updated at each iteration t of the algorithm, is initialized with all the signals available ($L^{(0)} = \hat{X}$)

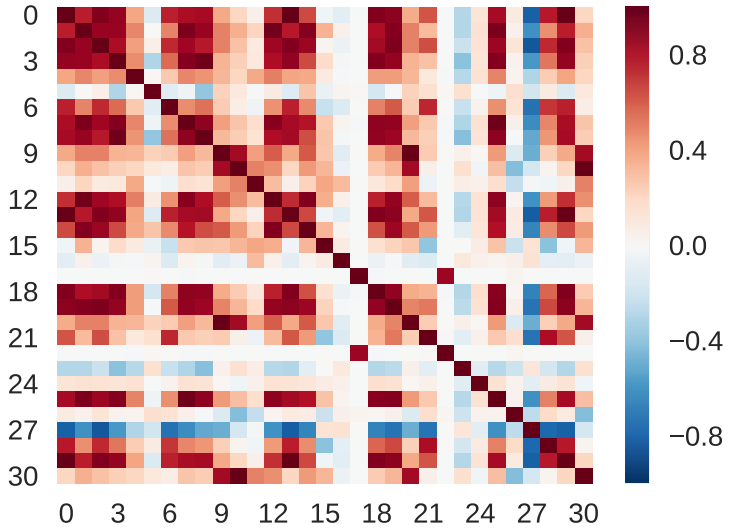


Fig. 5.8 Heatmap for the correlation matrix for a given cycle. The variable names have been replaced with numbers for visualization's sake

3. The set of "representative signals" R is initially defined as an empty set ($R = \{\}$); it will contains the final set of signals considered representative of the entire set $\hat{X}$ (i.e. $\bar{X}$ with the previously adopted notation).

4. The set of "discarded signals" D is initially defined as an empty set ($D = \{\}$); it will contain the set of signals that have been discarded during the signal selection process.

5. For each signal i in $L^{(t)}$, the sum of squared correlation coefficients s_i is computed: $s_i = \sum_{j \in L^{(t)}} r_{ij}^2$.

6. The signal $b = \arg\max_{i \in L^{(t)}} s_i$ is chosen from $L^{(t)}$ as the most representative of the signals left: $R = R \cup \{b\}$.

7. The set of signals $C^{(t)} = \{v \in L^{(t)} \ : \ |r_{vb}| \geq r_{min}\}$ contains all signals in $L^{(t)}$ that have a correlation with b larger than r_{min}. These signals are removed from $L^{(t+1)}$, since they are well represented by b: $L^{(t+1)} = L^{(t)} \setminus C^{(t)}$. We additionally store the discarded signals in $D = D \cup C^{(t)} \setminus \{b\}$.

8. If $L^{(t+1)}$ is empty the algorithm terminates, otherwise it continues with Step 5.

The algorithm is guaranteed to converge: the set $C^{(t)}$ is guaranteed to be non-empty, since it always contains at least $\{b\}$ – as such, we are always removing at least one element from $L^{(t+1)}$, when starting from $L^{(t)}$.

It can be easily shown that the following two properties hold for CORR-FS:

- *All selected signals in R will have correlation, among one another, smaller than r_{min} in absolute value* (i.e. there is limited redundancy in the signals kept). In fact, for any pair of signals $a, b \in R$ added at times t_a and t_b (assuming $t_a < t_b$ without losing on generality), if $|r_{ab}| \geq r_{min}$, by construction $b \in C^{(t_a)}$, which implies that signal b will be discarded from $L^{(t_a+1)}$. It follows that b can no longer be selected to be added to R.

- *All discarded signals in D have a correlation greater than r_{min} with at least one signal in L* (i.e. each discarded signal is "represented" by at least one of the signals kept). In fact, if $a \in D$, it means that a time t_b exists where $a \in C^{(t_b)}$. By construction of $C^{(t_b)}$ there exists a signal b, chosen at time t_b, such that $b \in R$ and $|r_{ab}| \geq r_{min}$.

All comparisons made against the minimum threshold r_{min} are done in absolute value to also consider negative correlations (which represent strong correlations nonetheless). The value r_{min} is the only hyperparameter of the proposed algorithm: it is a value in the range $[0, 1]$ can be easily tuned through either a grid search, or based on the domain-driven constraints that may be in place. The sum of squared correlation coefficients is used when computing s_i (Step 5) to reduce the influence of correlations close to 0 (alternatively, the absolute values could have been summed instead, if more weight was meant to be assigned to small correlations).

We note that CORR-FS is completely unsupervised: as such, it is suitable for identifying useful signals regardless of the final target. This is particularly valuable since it makes the selection process independent of any labels that may either not be available, may be available in scarce quantities, or may be unreliable. For these reasons we find the algorithm to be a valuable contribution to this work of thesis. To assess the quality of the proposed signal selection approach, CORR-FS will be compared to various supervised and unsupervised signal selection techniques that are presented below. We do not consider techniques that compute new signals (e.g. linear combinations of existing signals) due to the constraint imposed by the domain

experts on the preservation of the original signals, to not hinder their understanding of the process.

Unsupervised approaches

Unsupervised signal selection algorithms find the best subset of signals by analyzing the hidden structure in unlabelled data [100]. These algorithms exploit solutions that leverage the correlation or similarity among signals [101] to reduce redundancy in the data, or data transformations (e.g. PFA [102], PCA [103]) to identify combinations of signals that primarily represent the phenomenon under study. For this thesis, we compare two approaches that we briefly describe below, plus the previously introduced CORR-FS.

Feature Similarity (FSFS) [101] uses a metric called *maximal information compression index* to reduce redundancy in the dataset. This algorithm requires a single parameter k representing the desired reduction. For each value of k, the algorithm returns a different subset of signals. In short, it produces the optimal subset of signals whose amount of information is k time lower than the original complete dataset. We identify the best subset of signals by searching which value of k maximizes the *Representation Entropy* [101]. This metric represents how equally the information is distributed among the signals.

Principal Feature Analysis (PFA) [102] exploits an algorithm based on the *Principal Component Analysis* (PCA) [104] and the *k-means* clustering algorithm [105] to identify the subset of signals retaining most of the dataset information. This algorithm requires two parameters: p, the number of components used by the PCA to represent each signal x_i, and k, the desired number of clusters computed by k-means. The algorithm selects only one signal for each cluster; hence, k also represents the number of signals selected at the end of the selection process. We identify the best value of p by evaluating the knee point between the number of components used by the PCA and the cumulative explained variance, i.e. the total amount of dataset variance represented by those components. Then, we find the best value of k by optimizing the clustering quality metrics, i.e. optimizing the *SSE* (sum of squared errors) or the *silhouette* index [106] which quantify the cohesion of points within

each cluster at the end of the clustering process (as well as how separated clusters are, for the silhouette).

Supervised approaches

Supervised signal selection differs from the above approaches since we use a classification algorithm to select the subset of the most important signals. In short, we train different classifiers, providing a different subset of the signals. The main advantage of these solutions is that they evaluate the combined predictive capabilities of the input variables and optimize the choice for each classifier. However these approaches are typically more computationally intensive, given the need to build a complete pipeline. Additionally, they produce choices based on the specific task at hand. Changing tasks implies extracting possibly different sets of signals.

To use these algorithms, we transform each signal $x_i \in \hat{X}$ into a set F_i of n features, such that $F_i \in \mathbb{R}^n$. We provide details about the transformation in the next subsection. Then, each algorithm can provide information about the importance of each feature separately, i.e., allowing us to rank signals. We use this information to reconstruct the importance of each signal and use this importance to select the subset of the most important signals. Here we consider the following classifiers:

Random Forest (RF) [107] exploits a ranking algorithm based on how useful each feature is within the trees learned by the forest. These feature importances (FI) can be computed for each feature based on the total contribution in terms of impurity decrease that the specific feature as brought, throughout all trees. To extract it, we build a model by using all cycles D and all features F_i derived from $x_i \in \hat{X}$. Since multiple features F_i are extracted from each signal x_i, we compute the signal importance (SI) as the sum of all the feature importances of the signal's features as $SI(x_i) = \sum_{j=1}^{n} FI(F_i(j))$.

Finally, we rank the signals according to their SI and select the best subset of signals by using the knee point identification proposed by [108].

Random Forest - Recursive Feature Elimination (RF-RFE) [109] is based again on the signal importance but recursively eliminates the least important signals considered during the training phase. We start from a set of signals $S^{(0)} = \hat{X}$. Then,

we iteratively train a model by using all cycles and features F_i of all signals $x_i \in S^{(t)}$. At the end of the training phase, we record the classification performance of the t^{th} model. For each signal in $S^{(t)}$ we compute the signal importance as described before, and discard a subset $D^{(t)} \subset S^{(t)}$, identified as the least important signals ($|D^{(t)}| = \lceil \eta |S^{(t)}| \rceil$, $\eta \in (0,1)$ represents the fraction of discarded signals). The new set of signals is computed as $S^{(t+1)} = S^{(t)} \setminus D^{(t)}$. We build a new model and iterate until the remaining set of features is empty. We use the out-of-bag (OOB) error to quantify the quality of each subset of signals. We select the set of signals having the lowest OOB error.

SVM - Recursive Feature Elimination (SVM-RFE) [110] instead leverages a Support Vector Machine (SVM) classifier. Similarly to random forests, SVM returns at the end of the learning process the importance of each feature utilizing the feature weights learned. A similar process to the one used for random forests is set in place for the iterative feature elimination process.

Windowing

Each signal belonging to each cycle is a time series that assumes continuous values. The length of each time series corresponds to the length of the entire cycle, i.e. 3750 seconds. However, we may be interested in studying smaller time windows: it is unclear, from a data-driven perspective only, whether it would be more reasonable to have longer or shorter windows. Arguments can be made for both situations: longer windows imply having more data to make decisions off of, whereas shorter windows may allow detecting local trends that may get lost when considering longer sequences (as can be the case with all memory-based approaches, e.g. Recurrent Neural Networks).

Because of this, we consider the option of splitting each cycle into multiple non-overlapping sub-cycles of duration ΔT. The length of ΔT is one of the additional hyperparmeters to be tuned since, as already argued, there is no unambiguous choice for it.

Feature extraction

Given a window of length ΔT and the set of selected signals $\bar{X}$, we consider different strategies to extract a set of features F_i for each signal $x_i \in barX$. We consider well-established methods for time series, and propose a specific methodology specifically tailored to the characteristics of our problem.

Time Series Feature Extraction (TSFEL) [111] extracts more than 60 features from the original time series, including the statistical, temporal and spectral characteristics of each signal.

Automatic Feature Engineering for Forecasting (VEST) [112] employs several steps to extract features out of time series data: it groups observations in batches, summarizes each batch with statistical characteristics, and then returns the most relevant features according to a ranking criteria.

Tsfresh [113] offers 63 time series characterization methods, including continuous wavelet analysis, fast Fourier transform, time series length, mean, max, and median, etc., to extract up to 794 time series features out of each time series.

Ad-hoc Guided by the rationale that the clogging process is very slow, and that we are willing to compute simple features that could be computed on board, we define an *ad-hoc* strategy that summarizes each signal x_i using statistics of samples belonging to a time window. More specifically, we compute:

- *Mean value*, since the clogging of the oxygen sensor may introduce an offset on a signal which is proportional to clogging;
- *Standard deviation*, since the clogging of the oxygen sensor may affect the variability in recorded signals;
- *Percentiles*: the percentiles summarize the cumulative distribution function (CDF) of the signal values over time. They allow the system to identify those phenomena that change the signal values distribution.

To preserve information about the behavior of signals in time, we compute the same features for the discrete derivative $x_i'(t) = x_i(t) - x_i(t-1)$ of each signal x_i. This allows us to capture the variability of the signal over short time intervals, providing useful information about how quickly the signals change. By considering both the distribution of the signal values and the variability of the derivative, we can gain a more complete understanding of the signals.

Feature selection

After the feature extraction process, each signal is represented by f features: since these features are extracted from the same signals, it may happen that some features could be highly correlated, despite the initial signal selection process. For example, mean and 50^{th} percentile may convey very similar information for symmetrical distributions. This redundancy results in a dimensionality of the problem that is possibly larger than then one strictly needed.

To mitigate the curse of dimensionality problem [114] and improve the predictive performance of the proposed pipeline we perform the following feature selection process. All features are sorted according to their feature importance in a random forest model. We iteratively train classifiers by adding a new feature to the pool of available ones, and evaluate the classifiers based on commonly adopted metrics (e.g. F_1 score, accuracy). We repeat the process until all features have been added. We choose as the final set of features the one for which any further addition does not prove to be beneficial in terms of improvement in performance.

The initial order used to sort the features is a heuristic that we introduced to significantly limit the number of sets of features to be tested. In this way, we only need to assess $|F|$ features (F being the set of all available features). This is in contrast with $2^{|F|}$, the total number of sets that could potentially be obtained from the original F features, which is obviously intractable for any F outside of toy examples.

Historicization

The intuition suggests that we may improve the performance of the final model by considering not only the information of the current window, but also of previous

time windows. Therefore, we consider adding to the pool of features the features extracted from the past h_0 windows.

While historicization can improve the model's performance, it also increases the number of features to consider, thus falling back to a problem where high redundancy may be present. We thus use an additional feature selection step (as previously described) to identify the most important features for model training after the past windows are included.

5.6 Model training

There are many different classifiers in the literature, each with its own strengths and weaknesses. Based on the preliminary results shown in [115], we identify a fully-connected neural network (or multi-layer perceptrons - MLPs) [116] as being a reasonable model for the task at hand. We have additionally considered other models, e.g. decision trees [117], random forests [118], and support vector machines [119]. The overall performance of these models was lower w.r.t. that of MLPs and will not be reported in this thesis for brevity. However, some considerations will be made for the decision tree in Section 5.9.

To find the combination of hyperparameters that maximizes performance for the proposed classifiers, we use an extensive grid search that builds and assesses the performance of multiple models, each trained and evaluated on a separate set of hyperparameters. By doing this, we are able to identify the combination of hyperparameters that yields the best results on unseen data.

Model Validation

In this study, we consider two techniques for validating our predictive maintenance pipeline: traditional k-fold cross validation and time series cross validation.

k-fold cross validation is a commonly-used technique, particularly when only limited amounts of data are available, as is the case here. It consists in splitting the dataset into separate partitions, or folds, and iteratively use all but one of them for training, and the remaining one for validation. However, this kind of approach assumes that the collected data points are all independent of one another. This

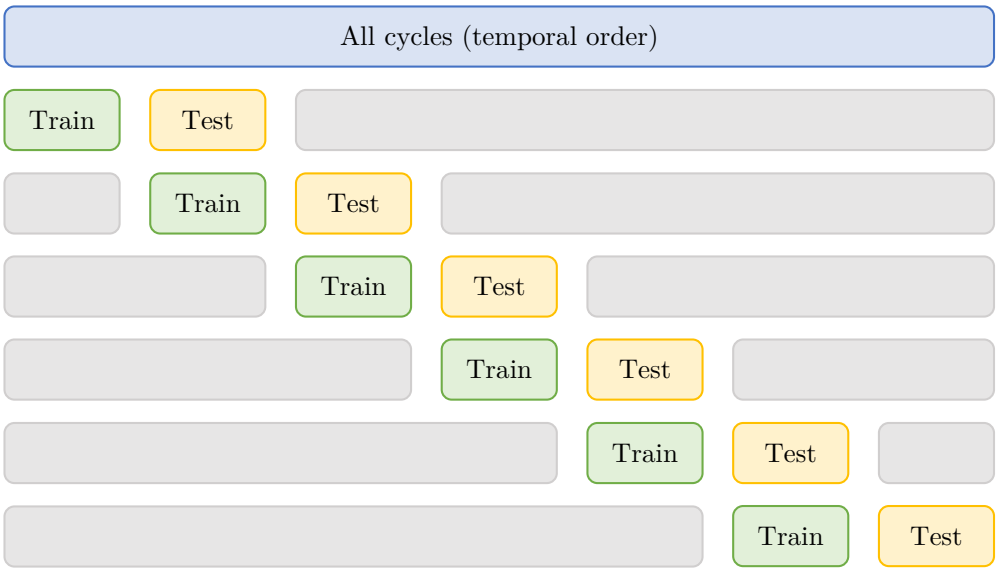


Fig. 5.9 Time series cross-validation: training and test sets change through time. Test data always follows training data in time to prevent data leakage. The data in gray is left unused for the specific step.

may not necessarily be the case for this problem for two reasons: the windows extracted from the same cycle share the same pseudo-label, and adjacent cycles may possibly share the same label, due to the assumption of continuity of the observed phenomenon. Therefore, having window/cycle i in the test set and window/cycle $i+1$ in the training set may be seen as a case of data leakage. We can easily address the "windows-in-cycle" problem by guaranteeing that all windows belonging to a cycle fall within the same fold. However, the inter-cycle relationships cannot be easily solved, since the temporal relationship is embedded in the problem we are handling.

Because of this, we additionally use a time series cross validation, which is designed to specifically address this problem. In this technique, we use a given sequence of cycles for training and the following ones for testing. We then move along time changing training and test data, as exemplified in Figure 5.9.

While this approach is more rigorous, our data-scarce scenario makes this approach wasteful: a large fraction of available data is not used for training nor testing. On top of being wasteful, the models learned will be trained with lower amounts of data than theoretically possible: we can expect those models to generalize poorly

due to plausible overfitting. The alternative time series validation option consists in iteratively increasing the training set size with each iteration (i.e. the left-most data in Figure 5.9 becomes part of the training set). This partially addresses the aforementioned problems, but without fully solving them and creating new ones: for example, different models would now be trained on training sets of different sizes, thus making comparisons even harder.

Because of these problems, we adopt a hybrid approach that uses both k-fold (making an assumption of independence) and time series cross validations. We perform the harmonic means of the performance metrics extracted to obtain a single value that represents the overall model behavior. We expect an under-estimate of the performance from the time series validation and an over-estimate for the k-fold validation. The harmonic mean is thus expected to balance these two quantities.

For k-fold cross validation we use a number of folds $k = 10$, whereas for time series cross validation the train and test sizes are set to 100 cycles, with a step of 3 cycles.

The following metrics are used for the evaluation of the model:

- *Accuracy*: this provides an overview of the overall performance of the model. While not our main metric of interest, it offers a valid "sanity check" metric, to guarantee that the model is not overly degraded when trying to optimize its behavior for the class of most interest (i.e. the red class).
- F_1 *score for the red class*: we are mainly interested in studying the performance of the model in terms of the red class, i.e. the class that, when detected, will produce the most important alert. Since both high precision and recall are desired, the F_1 score is considered as the main metric of interest. For completeness, precision and recall will also be reported.

5.7 Experimental results

Experimental validation has been carried out to assess PREPIPE's performance in terms of predicting capabilities and to offer support to the identification of the best hyperparameters for the pipeline.

	Algorithm	Input Parameter	Optimization	# Selected Signals
Unsupervised	CORR-FS	r_{min}	Knee Point	13
	FSFS	k	Representation Entropy	5
	PFA-SSE	p, q	Knee Point	12
	PFA-Silhouette	p, q	Silhouette	10
Supervised	RF	RF-Config	Knee Point	9
	RFE-RF	RF-Config	OOB Error	2
	RFE-SVM	SVM-Config	F_1 score (red)	4

Table 5.3 Signal Selection.

The unlabelled dataset resulting from the process described in Section 5.5 and the labels identified by the process in Section 5.4 are merged together to create the final, labelled dataset. As already explained, the class pseudo-label assigned to each cycle is either one of the labels “red”, “yellow” or “green” based on the PREPIPE’s labelling process.

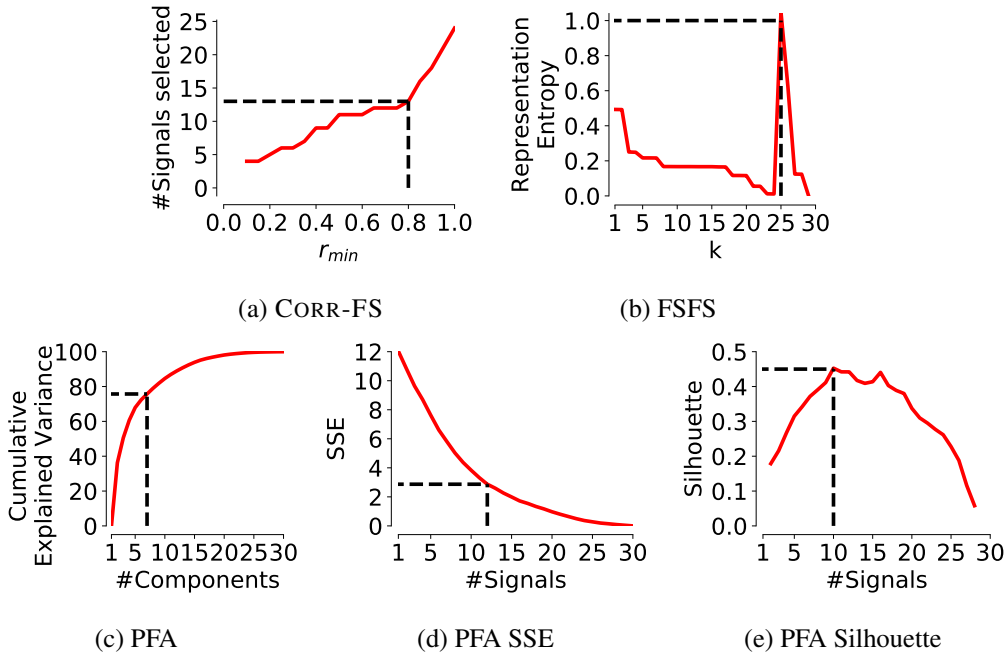
PREPIPE has currently been implemented in Python, using scikit-learn [56]. Most experiments have been performed on a dedicated server running Ubuntu 16.04, with 12 cores at 2.67 GHz and 32 GB of main memory. To speed-up the hyperparameter selection process, we parallelized the grid search on a larger cluster comprised of 36 nodes. The code is available at [120].

Impact of signal selection

In this section, we compare the various signal selection approaches considered in terms of classification performance. We consider both unsupervised and supervised approaches to identify the optimal subset of signals for modeling the clogging status of the oxygen sensor. The details of each algorithm, including the parameters and optimization criteria, are summarized in Table 5.3. The number of selected signals at the end of the process is also reported.

Unsupervised approaches

We can exploit the full dataset D for the unsupervised algorithms since we do not need to run the whole pipeline with these algorithms. In the following, we describe how we proceed with each algorithm.



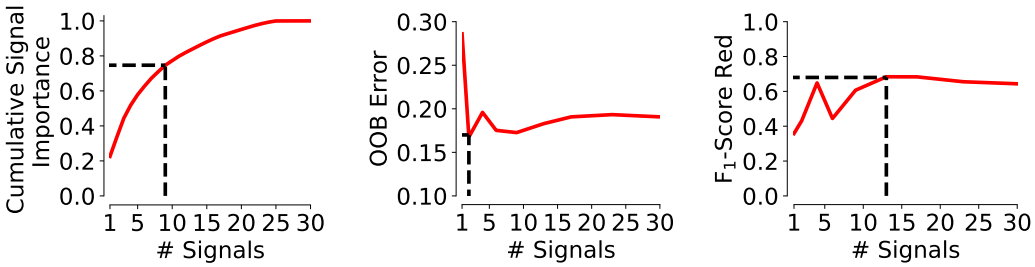
(a) CORR-FS (b) FSFS

(c) PFA (d) PFA SSE (e) PFA Silhouette

Fig. 5.10 Identification of the best number of signals with unsupervised algorithms.

Correlation-Based Feature Selection (CORR-FS). As already argued, the CORR-FS algorithm for the unsupervised signal selection procedure identifies one main hyperparameter of interest, r_{min}. Since the CORR-FS algorithm takes the absolute value of each correlation coefficient (because negative correlations are correlations nonetheless), it makes sense to only analyze values for $r_{min} \in [0,1]$. Intuitively, $r_{min} = 0$ implies that the lowest number of signals is selected (possibly only 1), given that each selected signal represents all others. By contrast, setting $r_{min} = 1$ results in the largest number of signals selected, as each selected signal only covers for the completely correlated ones.

The selection of this value requires finding a trade-off between the number of selected signals and their ability to represent the discarded variables well. The value can be selected either by setting an *a priori* constraint on the desired representativeness of the selected signals, or by studying the evolution of the number of selected signals as the coefficient changes. We adopt the latter approach to preserve domain-agnosticity. The value for r_{min} is found by identifying the knee point of the r_{min} vs number of selected signals plot (Figure 5.10a). We identify $r_{min} = 0.8$ as being the most suitable value, for a total of 13 signals being selected.



(a) RF (b) RFE-RF (c) RFE-SVM

Fig. 5.11 Identification of the best number of signals with unsupervised algorithms.

Feature Similarity (FSFS). Based on the implementation proposed in [121], we run the selection algorithm by concatenating all cycles into a single, longer one, then varying the parameter k. For each value of k, we extract the respective subset of signals and compute the representation entropy (RE), as suggested in [101]. Figure 5.10b shows the RE value as k increases. From this plot, we identify the maximum RE when $k = 25$, corresponding to the selection of 5 signals.

Principal Feature Analysis (PFA). Similarly to the FSFS algorithm, we concatenate all signal cycles to obtain a single time series for each signal, which is then normalized to have zero mean and unit variance. Finally, we run the PFA algorithm to identify the best number of components p representing the dataset. Figure 5.10c reports the cumulative explained variance as the number of components increases. We find 6 as the optimal number of dimensions to represent the majority of the dataset information, based on the knee point of the curve. As the last step, we run the k-means clustering algorithm to select the subset of signals. We choose k by optimizing either the SSE or Silhouette scores. Figure 5.10d and Figure 5.10e reports the trend of the SSE and Silhouette scores, respectively. We find two suggested signal subsets composed of 12 and 10 signals, respectively.

Supervised approaches

In this subsection we assess the performance obtained for the previously presented supervised signal selection approaches.

Random Forest (RF). We train a Random Forest configured with hyperparameters as suggested in [107]. We train it using the whole dataset and extract all features mentioned in Subsection 5.5 for all 30 signals. At the end of the training, we compute the signal importance for each signal. Figure 5.11a reports the signals ordered by their SI and the value of the cumulative signal importance. The knee point identification suggests selecting the first 9 signals as a possible subset.

Random Forest - Recursive Feature Elimination - RFR-RF. In this case, a separate random forest is trained at each iteration by using the full dataset and all features from the signals available in the current set (initially all of them). We use 2,000 estimators, as suggested in [109]. At each iteration, we discard the 20% least important signals according to their signal importance. Figure 5.11b reports the value of the out-of-bag error for the number of remaining signals. Here we select the subset having the lowest error (which amounts, in this case, to a selected subset of just 2 signals).

SVM - Recursive Feature Elimination - RFE-SVM. Similarly to the previous algorithm, we remove 20% of the least important signals according to the signal weight at each iteration. At each iteration, we evaluate the classification performance of each model with a standard 10-fold cross validation by using all cycles available and the features derived from the retained signals. Figure 5.11c reports the trend of the best F_1 score for the red class with a different number of signals. The performance is maximized when a subset of 4 signals is considered.

Benefits of Signal Selection

To assess the quality of each signal selection alternative, we build a full pipeline (all other parameters are set to some default values). We assess the quality in terms of accuracy and F_1 score computed using both k-fold and time series validation (reporting the harmonic mean between the two quantities, as already discussed).

Figure 5.12 shows the results for each signal selection technique, as well as for the baseline solution of not using any signal selection process. It can be observed that most of the proposed approaches do not result in a particular benefit in the overall pipeline w.r.t. not discarding any signal. CORR-FS and RF are the only

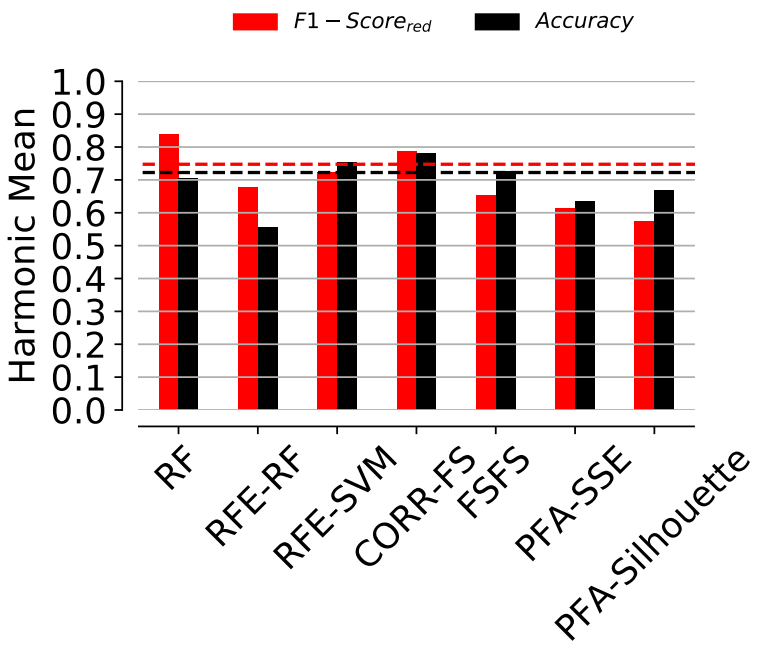


Fig. 5.12 Performance on a multi-layer perceptron for various signal selection techniques. The metrics reported are the harmonic means across 10-fold and time series validation. Dashed lines represents results without signal selection.

approaches for which some benefit occurs. Among these, only CORR-FS results in an improvement of both metrics, although RF achieves the largest overall F_1 score for the red class. We choose CORR-FS in that it presents the most consistent results across the two metrics. Additionally, the algorithm's unsupervised nature makes it more suitable for a wider set of situations.

Windowing

In this section, we investigate the impact of various windowing policies by varying the ΔT parameter. To do this, we divide each cycle, which is approximately 60 minutes long, into independent (i.e. non-overlapping) time windows of varying duration ΔT. These windows range from 60 minutes (one window per cycle) to 2 minutes (30 windows per cycle). We label each window with the pseudo-label of the cycle to which the window belongs: this makes the implicit assumption that the clogging status is consistent throughout the entire cycle, as already discussed.

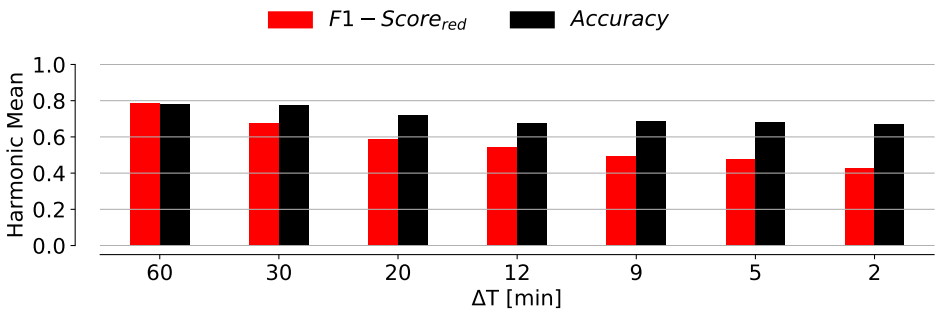


Fig. 5.13 Performance obtained with various window sizes.

Figure 5.13 reports the MLP classifier performance using, as usual, the harmonic means of F_1 score and accuracy as the window size changes.

A clear trend emerges: the shorter the cycle, the worse the performance. This trend is particularly evident for the F_1 score of the red class, which degrades significantly when using shorter time windows to extract features. The accuracy, on the other hand, has a small degradation in performance but plateaus for windows smaller than 12 minutes.

For this specific case, the best course of action is to monitor the clogging status every 60 minutes, indicating that the clogging phenomenon is better observed on longer time scales.

This result is also beneficial for deployment, as it allows the ECU to transmit less data and reduces the number of classification decisions that need to be made in the cloud.

Feature Extraction

In the next step, we move on to feature extraction. Starting from the 13 signals collected with a $\Delta T = 60$ minutes, we extract one set of features for *tsfresh*, a second set for *VEST*, four sets for *TSFEL*, and one set for *Ad-hoc*. For TSFEL, we consider (i) *All* the possible features, (ii) *All-corr* all those features that are not strongly correlated, (iii) *Statistical* features only, and (iv) *Temporal* features only.

Table 5.4 reports the number of features extracted by each strategy. Since the *tsfresh* library returns more than 10,000 features, we discard those results due to the

small number of experiments available, which would make the convergence of the training complicated.

Figure 5.14 summarizes the performance obtained by each strategy using a MLP classifier after grid search. Interestingly, the variety of features provided by the Vest and TSFEL packages does not help to describe the clogging phenomenon. The higher the number of features, the lower the performance compared to the Ad-hoc features, which consistently outperform the other strategies. We therefore consider the Ad-hoc feature extraction strategy in the following analyses.

	Approach	#Features
	Ad-hoc	286
	Tsfresh	10231
	VEST	640
TSFEL	All	5070
TSFEL	All-corr	3964
TSFEL	Statistical	468
TSFEL	Temporal	234

Table 5.4 Number of features per feature extraction strategy.

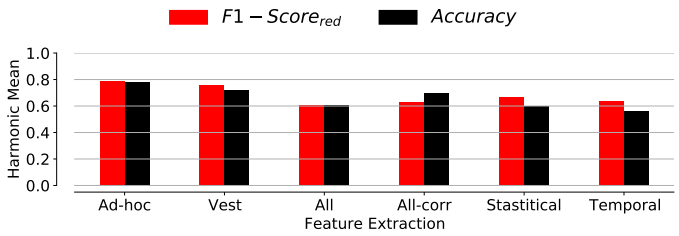


Fig. 5.14 Performance per feature extraction strategy.

Feature Selection

Figure 5.15 reports the performance achieved by a multi-layer perceptron network when trained on each subset of features, obtained according to the process explained in Section 5.5.

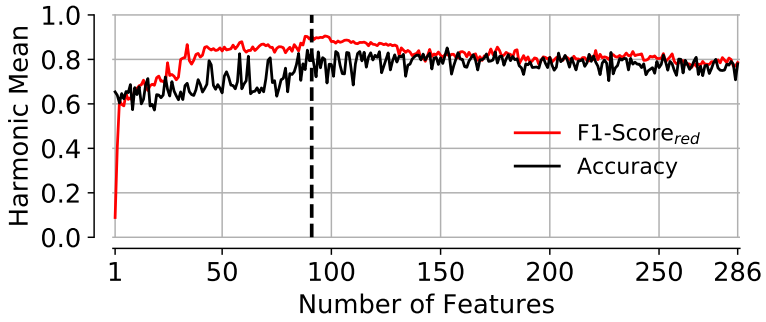


Fig. 5.15 Performance improvement due to feature selection.

It comes to no surprise that a low number of features results in poor model performance. However, the performance rapidly increase as new features are added:

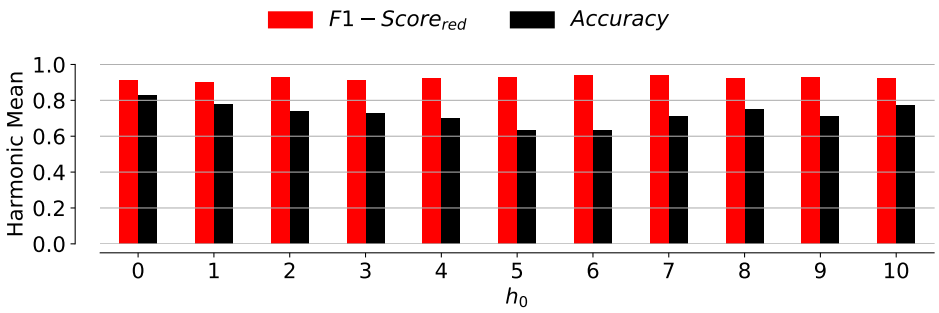


Fig. 5.16 Performance improvement brought by historicization.

this sharp increase is partially due to the ordering policy used, which chooses important features (as computed through a random forest) first.

We obtain the best performance on the red class with $\approx$ 85-95 features. A larger number of features results in a loss in performance in terms of F_1 score and no additional benefit in terms of accuracy. We thus select the number of featuers $\in [85, 95]$ that maximizes the accuracy: 91. This brings the number of features to one third of the original number (286).

Historicization

Finally, we evaluate the impact of including past information. For this, we enrich each time window with features from the previous h_0 time windows. We next run an additional feature selection step to remove any newly added redundancy. We train, tune and validate a MLP model to find the best possible performance for each historicization window.

Figure 5.16 reports the observed results. We observe minor changes in the F_1 score when increasing h_0 whereas the accuracy varies inconsistently: it decrease as h_0 grows to 5, to then increase again. Despite the slight increase in performance in F_1 score, the behavior in terms of accuracy is significant of the fact the performance on the other classes deteriorates. Introducing additional input data is thus not beneficial to the proposed pipeline – we therefore decided to use $h_0 = 0$ (i.e. no historicization is used).

Classifier	Parameter	Values
MLP	Input layer	$\|F\|$
	1^{st} hidden layer	{2,5,11,17,19,23,27,40,46,54}
	2^{nd} hidden layer	{2,5,11,17,19,23,27,40,46,54}
	Output layer	3
	Activation	{logistic, tanh}
	Solver	Adam
	Tolerance	10^{-4}

Table 5.5 Grid Search hyperparameter configuration.

Step	F_1 sscore			Accuracy
	Green	Yellow	Red	
Original	0.78	0.62	0.75	0.72
Signal selection	0.82	0.72	0.79	0.78
Windowing	0.82	0.72	0.79	0.78
Feature extraction	0.82	0.72	0.79	0.78
Feature selection	0.83	0.81	0.91	0.83
Historicization	0.83	0.81	0.91	0.83

Table 5.6 Recap of the best performance found at each optimization step.

Model training and tuning

As already discussed, we focus our study on the identification of a fully-connected neural network (or MLP) as the main model of interest. For the identification of the best hyperparameters we focus on the works presented in [122, 123]. Table 5.5 details the ranges we used for each of the main hyperparameters under study.

Overall performance

To summarize the contribution of each step in the optimization pipeline, Table 5.6 presents the results obtained throughout the pipeline by training a multi-layer perceptron classifier. We additionally include the F_1 score for the green and yellow classes. The two steps that bring the highest benefit to the pipeline are the signal and feature selection steps.

We do not observe any degradation in performance for the other classes: this implies that the choices made are not overly fit on the red class. This is useful in terms of identifying early symptoms of potential upcoming problems.

5.8 Comparison with deep learning methodologies

In the last decade, deep learning (DL) methodologies have gained momentum, thanks to the increase in computing capabilities and data availability, and have led to breakthroughs in many machine learning tasks [124]. Indeed, deep learning methodologies have been helpful in several fields such as image classification [125, 126], time-series prediction [127], and prognostics as well [128, 1]. Such increase in popularity is driven by the capability of deep learning solutions to abstract the data without complex manual feature engineering [124], and their good performance, e.g., high accuracy in classification problems.

Due to the heavily preprocessing-oriented nature of this work, we believe that the most useful comparison should be against deep learning techniques, to automate most of the work done this far.

We start from the initial set of 34 signals, i.e. the 30 signals that have been retained after the domain-driven selection, plus an additional 4 that were removed in the proposed pipeline because correlated with a coefficient of 1 with other signals. We decide to keep all 34 signals, and not just 30, to assess how well DL methodologies behave off the shelf, with as little feature engineering/selection as possible. We apply a light preprocessing step to normalize the signals using a z-score normalization and split our data following two alternative approaches:

- *Training on whole cycles (Whole)*: we use each of the 388 cycles as a separate input for the model, i.e., we train the model using the entire cycle at each step.
- *Time windows (Windowing)*: as in Sec. 5.5, we divide each experiment into independent time windows, setting $\Delta T = 100$ seconds, resulting in 14,356 inputs. Then, we feed each window separately to the model. Each window is labelled with the same cycle label it belongs to (with the previously made assumptions).

Finally, we validate the pipeline following the same approach adopted for the proposed methodology. Next we propose the two models used for this comparison, namely Convolutional Neural Networks [58] and LSTMs [129].

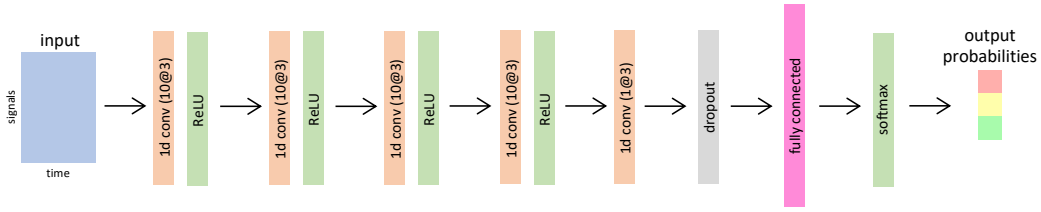


Fig. 5.17 Convolutional neural network architecture based on [1].

Classifier	Parameter	Values
CNN	# of conv layers	5
	kernel size (all conv layers)	3
	stride (all conv layers)	1
	output channels (conv layers 1-4)	10
	output channels (conv layer 5)	1
	dropout probability	0.5
LSTM	hidden state dimensionality	32
	# of recurrent layers	1

Table 5.7 Configuration used for the proposed deep learning architectures.

Convolutional Neural Network

In the prognostics field, authors in [1] proposed to use a Convolutional Neural Network (CNN) architecture to approach a remaining useful life (RUL) task for a turbofan engine degradation problem. Here we use it as a first state-of-art architecture. Figure 5.17 reports the complete DL architecture we used, inspired from [1]. Table 5.7 highlights the main hyperparameters used for the training of the model. Since we tackle a classification problem instead of a regression one, we insert a fully-connected network with a softmax activation function as the head of the network.

Long Short-Term Memory

For the Recurrent neural networks (RNN), we use a different state-of-art architecture, i.e., a bidirectional long short-term memory (LSTM) [130] model, which introduces a gating mechanism for retaining and discarding information. The bidirectionality of the LSTM implies that two LSTMs are trained simultaneously, using the signal in positive and negative time directions [130]. This has been shown to provide

Architecture	Preprocess	F1-Score			Accuracy
		Green	Yellow	Red	
CNN	Whole	0.792	0.571	0.587	0.680
CNN	Windowing	0.846	0.784	0.562	0.791
LSTM	Whole	0.775	0.526	0.360	0.654
LSTM	Windowing	0.861	0.812	0.817	0.836

Table 5.8 Wrap-up of the performance based on preprocessing methodology and deep learning architecture.

better results in both regression and classification problems. Table 5.7 lists the hyperparameters used for the proposed LSTM model.

Results and discussion

Table 5.8 reports the results for each DL architecture and preprocessing approach. The CNN does not provide satisfactory results on the red class with either preprocessing approach. On the LSTM architecture we instead observe performance comparable to those obtained with our pipeline. The best results are obtained when the signals are windowed: it is well known that the backpropagation through time leads to problems (e.g. exploding or vanishing gradients) over longer sequences. Although LSTMs mitigate this problem, they do not necessarily solve it, as we can empirically observe on the longer sequences.

We believe that the low amount of available data is the main cause for the underperformance of the deep learning methodologies w.r.t. the proposed one. Additionally, the available data has very low variability (i.e. all cycles are piloted with the same track), thus preventing DL models to observe a wide range of scenarios and extract useful and generalizable features from them. Thus, in limited data scenarios, we can still observe an improvement in performance driven by significant manual preprocessing and feature engineering.

However, it should be noted that a much lower effort has been put toward building the LSTM model than the full PREPIPE pipeline. An important takeaway here is that a small price in terms of performance may be paid in exchange for a much simpler process that requires little domain expertise.

5.9 Oxygen signal features

As previously discussed, the oxygen signal has been removed from the set of available ones. The choice was made as a way to explore the relationships among other signals with the problem analyzed: in other words, the domain experts deemed interesting exploring how the rest of the engine is affected by a faulty oxygen sensor.

From a data-driven perspective it is instead interesting to explore whether the oxygen sensor collected from Program A can be of any use when predicting the target label.

We already know that, by construction, a part of the oxygen signal collected with Program B can be – and is – used to predict the clogginess of the engine. However, Program B is sampled a 320 times the sampling frequency of the oxygen signal from Program A. Measuring the response time from Program A does not produce useful results: the sampling frequency of 1 Hz makes the uncertainty of a measurement of values between 1 and 2 seconds (i.e. the range of values for the response time) too large.

However, it may be be the case that the oxygen signal across the entire cycle, although sampled at a lower frequency, may still carry information about the clogginess of the sensor.

We thus build a decision tree model to predict the target class, given as input all 13 signals plus the oxygen one (the rest of the pipeline is applied as already described). This very simple model achieves an accuracy of 0.80 and an F_1 score of 0.83 for the red class. These results are worse than the ones observed for the multi-layer perceptron but, considering the simplicity of the model, we can intuitively understand that the oxygen signal does carry useful information.

While we will not analyze the resulting decision tree in detail, we observe that the root node (i.e. the node where the "best" split on the overall dataset is identified) contains one of the percentiles (the 90^{th}) computed on the distribution of values of the derivative of the oxygen signal.

We thus additionally study the distribution of the 90^{th} percentile of the derivative of the oxygen signal. We show this result in Figure 5.18, for the 3 classes separately. This very clearly shows that red (clogged) cycles are characterized by lower derivatives (i.e. slower signals). The opposite is true for green cycles.

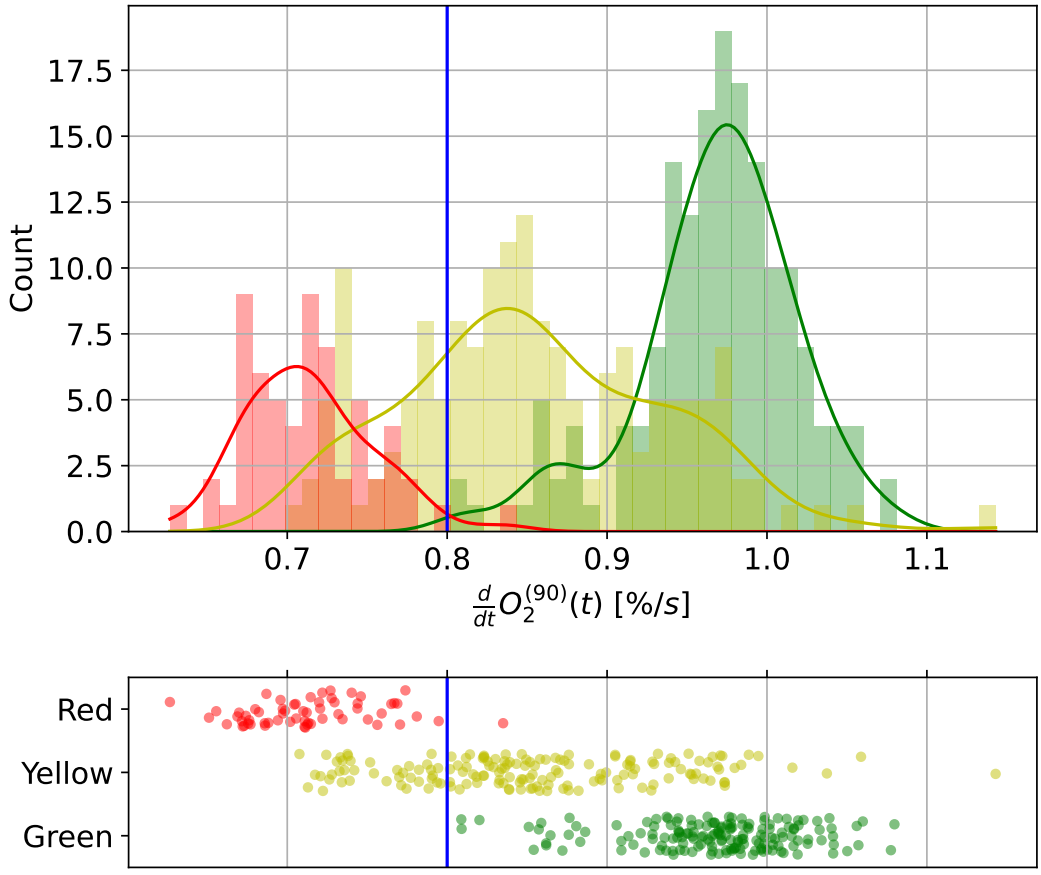


Fig. 5.18 Distribution of the 90^{th} percentile of the derivative of the oxygen variable, by class. The vertical blue line represents the splitting value identified by the decision tree.

We thus conclude that, while a lower sampling frequency prevents us from accurately measuring specific parameters (e.g. the response time), the overall distribution of values is still very helpful in addressing the sensor clogging problem. It should be noted that this conclusion can be easily achieved with the "right" feature engineering process. The raw signals would not have been as meaningful in quantifying this kind of behavior without the proper preprocessing.

5.10 Discussion

Part of this work of thesis has been focused on studying real scenarios that are affected by a lack of supervised data. To this end we presented PREPIPE, a pipeline aimed at addressing the oxygen clogging detection problem.

The pipeline is built on top of the proposed semi-supervised labelling technique: instead of relying on human annotations, we proposed extracting less reliable labels without any human effort. We reduce the noise in the assigned labels by studying their evolution through time.

The rest of the pipeline processes a large number of time series containing the sensors' data collected by the engine: we extensively evaluate each preprocessing step to optimize the predictive performance of the framework. Our results show how domain experts can take advantage of our framework to select the best subset of signals and features to predict the status of the system under analysis.

The other important contribution to this work of thesis is a correlation-based signal selection algorithm. This algorithm can be used to identify correlations among time series with no supervision required. This makes this approach relevant when labels are scarce, unreliable or missing.

Although not entirely aligned with the core goals of this thesis, we additionally covered other aspects of interest for this work. Among them, there are:

- The handling of time series data: we empirically assessed how a well-crafted feature engineering step can produce meaningful representations that can outperform those learned through feature learning (e.g. via LSTMs). We acknowledge that this fact is significantly related to the core problem of interest, and the results may vary for different kinds of tasks.

- The careful steps taken to guarantee that only a limited set of features is extracted and used. Since the final goal of this project is to deploy the system as a cloud-based service, communication between vehicle and servers is necessary. In a "default" scenario, all signals would have to be transmitted in near real-time. We instead identify a solution that allows only for a few features to be extracted and sent "in batches" (time windows): instead of sending 180,000 samples each hour (50 signals sampled at 1 Hz), we reduce the number of features necessary to $\tilde{1}00$ (or $\approx 99.9\%$ less data). All feature extraction steps are computationally inexpensive and can be computed on-board with little computing power.

The currently proposed pipeline has been developed using data collected in a test bench. The track used is meant to reproduce behaviors that are found in different settings (e.g. driving on a highway or in a city). However, the fact that the same track is used for all cycles implies that the variance observed is rather low. As a next step, we will consider collecting data from real vehicles driving in various conditions. With that data, it will be possible to understand the extent to which a model trained on test bench data can generalize to real driving conditions.

Being able to generalize a model to a different domain is, as already argued in the introduction, an effective way of handling label scarce scenarios: if enough data can be collected in a test bench and that data can be used to train a model that generalizes to different data distributions (e.g. people driving real vehicles), then the data collection cost is significantly reduced.

In a separate work carried out along with GM, we studied one such scenario, where the problem to be addressed was one about the conditions of the High-Pressure Fuel (HPF) system [18]. The adopted pipeline resembles the one proposed in this work on the oxygen sensor. However, the data was collected specifically for this study. As such, the ground truth was more easily enforced by the domain experts at data collection time.

This makes the entire pipeline a fully supervised one. The results obtained reflect this. Additionally, a second phase was carried out with data being collected from real vehicles being driven in different settings. Collecting this kind of data is even more expensive than collecting data in a test bench, given the necessity of having a human actually driving the vehicle the entire time (as opposed to programmatically

controlling the engine and its response). For this reason, the only data collected was used for validation, rather than for retraining.

The results obtained for that experiment show that a model trained on the standard cycles generalizes well to data collected on real vehicles. With cautious optimism, we can thus find it reasonable to expect that the same generalizability should also apply for the oxygen sensor problem.

Chapter 6

Conclusions

This work of thesis has been focused on the exploration of the field of machine learning in those situations where labelled data is scarcely available. We have already discussed many of the implications and future directions that could be pursued for each of the works presented thus far. In this chapter, we first reiterate the main reasons that have led to focusing this work of thesis on the topic of limited label learning. Then, we additionally summarize the main aspects of interest highlighted throughout the thesis and we identify some potential topics that may be worth investigating in the future.

6.1 Learning in label-scarce scenarios

It is a well-established fact that machine learning algorithms need significant amounts of supervision to achieve performance on par with that of humans. Achieving performance comparable with that of humans is obviously a tremendous effort. However, to get close to this target, large amounts of data are required: a popular example is the one given by ImageNet [131]. Human-comparable performance has been achieved relatively soon [132], but it should be noted that these models are trained on a million images, 1,000 for each class. As humans, we are typically able to learn from much smaller sets of data and can generalize significantly better than algorithms can. For this reason, the pursue of a label-scarce approach to machine learning is the next logical step to be taken.

A learning process that is closer to that of humans however is not the only reason why label-scarce learning is a task of interest. Limits in label availability are often dictated by the context where machine learning techniques are applied. As already argued throughout the thesis, acquiring human-labelled data is expensive. Thus, being able to produce well-performing models with very limited data available is also beneficial for practitioners, not only researchers.

6.2 A recap of the main contributions

In this thesis we covered topics that span various stages of the spectrum of label unavailability. From situations where labels are completely unavailable, to situations where labels are available but in a domain that differs from the one of interest, to situations where only a small subset of available data is labelled, with large amounts of unlabelled data to be additionally exploited.

The unsupervised setting is easily the hardest among all scenarios we have considered in this thesis. If no supervision whatsoever is available, the only task that can be defined is one of pattern identification, e.g. by clustering points based on shared properties. The validation of such patterns is non-trivial, given the lack of supervision. In some cases, this kind of validation has been referred to as a "black art" [133]. On top of this esoteric endeavor, another classical problem of clustering is given by the typical computational complexity that comes with it. We discussed how the majority of clustering techniques are either superlinear, non-incremental, or both. Among the few exceptions we identify self-organizing maps. We present in Chapter 3 an approach for making the training of self-organizing maps faster. This is done by first building a smaller model (i.e. a model with less representational capacity), then propagating it to a larger one, which is finally fine-tuned. We show the theoretical improvement in training time that such a technique can obtain over the traditional self-organizing map training, and we show that we indeed observe this improvement experimentally. Although it is non-trivial to formalize the degradation that the proposed approach has in terms of performance, we run extensive experiments to empirically assess this factor. The conclusions that we have reached is that, although there is a slight degradation in performance, it is typically an acceptable one, when balanced by the reduced computational effort required.

For domain adaptation, i.e. the task of learning from adjacent domains, we worked within the field of NLP. Starting from knowledge available in English (a typically resource-rich language), we aimed at propagating the available knowledge to a language where no prior knowledge was available. We do so by exploiting alignments of word embeddings that allow to find mappings between the two domains. After the mappings are found they are encoded as graphs: a gradient descent-based optimization is then used for the final knowledge propagation on the graph. We showed that this approach consistently outperforms techniques that are based on a fixed mapping between the source and target domain. We additionally noted that these fixed mappings between domains are also typically hard to obtain, especially for low-resource domains, which are the focus of this work. We drew some considerations based on the generalizability of the proposed method to scenarios outside of NLP: in short, the proposed propagation technique may be used in other scenarios where the same entities can be found across different domains (e.g. users in different social networks).

We additionally covered the topic of semi-supervised learning, where models are trained both on labelled and unlabelled data. A common trend in semi-supervised learning is that of building soft labels (or pseudo-labels) for unlabelled samples, especially for those for which the model has high confidence in its capability of assigning the correct class label. In this case, we focus on improving the confidence mechanism used by a state-of-the-art semi-supervised technique (FixMatch). Instead of using the implicit confidence that can be extracted from the predicted probability distribution, we introduce an explicit confidence output. We provide initial empirical evidence that the proposed confidence approach can improve the quality of the learned model, especially at early stages. However, we acknowledge that these results are still preliminary and that further investigations are required before more conclusive considerations can be drawn.

Finally, we thoroughly cover an applied case study. The setting is in the field of predictive maintenance, with the goal of predicting the clogging status of an oxygen sensor. We consider this case study of interest due to the scarce supervision that is provided: the available data is repurposed from a different experiment and thus lacks proper labels. We propose an approach to assign labels in a semi-supervised manner by exploiting domain knowledge, additional available data and some assumptions made on the distribution of the data. On top of that, we adopt a signal selection algorithm that is unsupervised in nature. This is particularly useful considering

that (i) the labelling is noisy and may thus prove to be unreliable and (ii) since the approach is unsupervised, it can be used in different settings with little changes. For the sake of completeness, we covered the entire data science pipeline introduced, although some aspects of it may not be completely aligned with the goals of this work of thesis.

6.3 Future directions

Overall, the research presented in this thesis has highlighted the potential of machine learning to address the challenge of label scarcity and has demonstrated the effectiveness of unsupervised, domain transfer, and semi-supervised learning approaches in this context.

However, there are still many open research questions and opportunities for further development in this area. One promising direction is to investigate the integration of multiple approaches, such as combining domain transfer and semi-supervised learning to address situations where only few labels are available, and only in an adjacent domain. In the field of NLP, for example, we may want to propagate information across languages, when little information is available even in the source language. This is, for example, applicable when we want to work on closely related and relatively unknown languages. A very relevant example is given by the Italian language and the over 30 languages within Italy that are currently endangered, according to Unesco [134]. Within this context, there has been recent interest in addressing the variety of Italian languages with NLP-based technique [135]. We believe this to be a worthwhile task, which we will consider addressing in the near future.

A different direction that has not been the focus of this thesis, but that is a valid and very relevant option, is the topic of active learning, in which the model actively selects which examples to label in order to maximize its learning efficiency. This could be particularly useful in scenarios where the cost of obtaining labels is high, as it allows the model to prioritize the most informative examples for labelling. In pool-based active learning (where a pool of all unlabelled points is available), the focus of the model is on identifying the samples for which acquiring a label would be most beneficial for the learning process [136]. A possible selection policy would be requesting labels for points for which the model produces the least confident

predictions: because of this, the discussion about confidence estimation made in Chapter 4 is also very relevant in this field. The main difference with semi-supervised learning would be that, instead of inferring the pseudo-labels for points that are most confidently predicted, with active learning the model would be able to request new labels for the points of least confidence.

Another interesting direction for future research is to explore the use of meta-learning to improve the learning conditions when labels are scarce. Meta-learning is concerned with the topic of "learning to learn", or adapting the learning process itself to a new task or domain. This could potentially enable models to adapt more quickly and effectively to new label-scarce scenarios, by leveraging their previous learning experiences to guide their learning strategy.

In conclusion, the research presented in this thesis is just a small contribution to the extremely large and complex field that is machine learning with limited label availability. We considered the problem from various angles and addressed it with several approaches. By continuing to investigate and refine these approaches, as well as by exploring new directions such as meta- and active learning, we aim at further improving the boundaries of what can currently be achieved in these scenarios that are so limiting and yet so ubiquitous.

References

[1] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, 172(C):1–11, 2018.

[2] Xin Dong and Gerard de Melo. Cross-lingual propagation for deep sentiment analysis. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 5771–5778, New Orleans, Louisiana, USA, 2018. AAAI Press.

[3] Juan C Rojas, Kyle A Carey, Dana P Edelson, Laura R Venable, Michael D Howell, and Matthew M Churpek. Predicting intensive care unit readmission with machine learning using electronic health record data. *Annals of the American Thoracic Society*, 15(7):846–853, 2018.

[4] Abhimanyu Roy, Jingyi Sun, Robert Mahoney, Loreto Alonzi, Stephen Adams, and Peter Beling. Deep learning detecting fraud in credit card transactions. In *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pages 129–134. IEEE, 2018.

[5] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[7] Ming Yin, Jennifer Wortman Vaughan, and Hanna Wallach. Understanding the effect of accuracy on trust in machine learning models. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–12, 2019.

[8] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[9] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[10] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

[11] Shiliang Sun, Honglei Shi, and Yuanbin Wu. A survey of multi-source domain adaptation. *Information Fusion*, 24:84–92, 2015.

[12] Flavio Giobergia, Luca Cagliero, Paolo Garza, and Elena Baralis. Cross-lingual propagation of sentiment information based on bilingual vector space alignment. In *EDBT/ICDT Workshops*, pages 8–10, 2020.

[13] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[15] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[16] Flavio Giobergia, Elena Baralis, Maria Camuglia, Tania Cerquitelli, Marco Mellia, Alessandra Neri, Davide Tricarico, and Alessia Tuninetti. Mining sensor data for predictive maintenance in the automotive industry. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 351–360. IEEE, 2018.

[17] Danilo Giordano, Flavio Giobergia, Eliana Pastor, Antonio La Macchia, Tania Cerquitelli, Elena Baralis, Marco Mellia, and Davide Tricarico. Data-driven strategies for predictive maintenance: Lesson learned from an automotive use case. *Computers in Industry*, 134:103554, 2022.

[18] Danilo Giordano, Eliana Pastor, Flavio Giobergia, Tania Cerquitelli, Elena Baralis, Marco Mellia, Alessandra Neri, and Davide Tricarico. Dissecting a data-driven prognostic pipeline: A powertrain use case. *Expert Systems with Applications*, 180:115109, 2021.

[19] Alan Ramponi and Barbara Plank. Neural unsupervised domain adaptation in nlp—a survey. *arXiv preprint arXiv:2006.00632*, 2020.

[20] Bing Liu. *Sentiment Analysis - Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015.

[21] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246, 2017.

[22] Hai Ha Do, PWC Prasad, Angelika Maag, and Abeer Alsadoon. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications*, 118:272 – 299, 2019.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[24] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751, 2013.

[25] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[26] Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[27] Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 959–962, New York, NY, USA, 2015. Association for Computing Machinery.

[28] Hitkul Jangid, Shivangi Singhal, Rajiv Ratn Shah, and Roger Zimmermann. Aspect-based financial sentiment analysis using deep learning. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1961–1966, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[29] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246, 2017.

[30] Erkin Demirtas and Mykola Pechenizkiy. Cross-lingual polarity detection with machine translation. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 9, Chicago, USA, 2013. ACM.

[31] Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. Multilingual subjectivity analysis using machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 127–135, Honolulu, Hawaii, USA, 2008. Association for Computational Linguistics.

[32] Kevin Duh, Akinori Fujino, and Masaaki Nagata. Is machine translation ripe for cross-lingual sentiment classification? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 429–433, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

[33] Xiaojun Wan. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[34] Yanqing Chen and Steven Skiena. Building sentiment lexicons for all major languages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 383–389, Baltimore, Maryland, USA, 2014. Association for Computational Linguistics.

[35] David Vilares, Carlos Gomez-Rodriguez, and Miguel A. Alonso. Universal, unsupervised (rule-based), uncovered sentiment analysis. *Knowledge-Based Systems*, 118:45 – 55, 2017.

[36] Gerard de Melo. Etymological Wordnet: Tracing the history of words. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1148–1154, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[37] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2nd edition, 2014.

[38] Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520, San Diego, California, June 2016. Association for Computational Linguistics.

[39] Flavio Giobergia and Elena Baralis. Reclaim: Reverse engineering classification metrics. In *2022 IEEE Fifth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 106–113. IEEE, 2022.

[40] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[41] Farhad Nooralahzadeh, Lilja Øvrelid, and Jan Tore Lønning. Evaluation of domain-specific word embeddings using knowledge resources. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

[42] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[43] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[44] I-Ting Chen, Li-Chiu Chang, and Fi-John Chang. Exploring the spatio-temporal interrelation between groundwater and surface water by using the self-organizing maps. *Journal of Hydrology*, 556:131–142, 2018.

[45] R Buchs, C Davis, D Gruen, J DeRose, A Alarcon, GM Bernstein, C Sánchez, J Myles, A Roodman, S Allen, et al. Phenotypic redshifts with self-organizing maps: A novel method to characterize redshift distributions of source galaxies for weak lensing. *arXiv preprint arXiv:1901.05005*, 2019.

[46] Fabian L Kriegel, Ralf Köhler, Jannike Bayat-Sarmadi, Simon Bayerl, Anja E Hauser, Raluca Niesner, Andreas Luch, and Zoltan Cseresnyes. Cell shape characterization and classification with discrete fourier transforms and self-organizing maps. *Cytometry Part A*, 93(3):323–333, 2018.

[47] Elia Oliver, Iván Vallés-Perez, Rosa-María Baños, Ausias Cebolla, Cristina Botella, and Emilio Soria-Olivas. Visual data mining with self-organizing maps for "self-monitoring" data analysis. *Sociological Methods & Research*, 47(3):492–506, 2018.

[48] Antonio Ciampi and Yves Lechevallier. Clustering large, multi-level data sets: an approach based on kohonen self organizing maps. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 353–358. Springer, 2000.

[49] Mu-Chun Su and Hsiao-Te Chang. Fast self-organizing feature map algorithm. *IEEE Transactions on Neural Networks*, 11(3):721–733, 2000.

[50] Mu-Chun Su, Ta-Kang Liu, Hsiao-Te Chang, et al. Improving the self-organizing feature map algorithm using an efficient initialization scheme. *Journal of Applied Science and Engineering*, 5(1):35–48, 2002.

[51] N Bandeira, VJ Lobo, and F Moura-Pires. Training a self-organizing map distributed on a pvm network. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, pages 457–461. IEEE, 1998.

[52] Daniel Garabato, Carlos Dafonte, Minia Manteiga, Diego Fustes, Marco A Álvarez, and Bernardino Arcay. A distributed learning algorithm for self-organizing maps intended for outlier analysis in the gaia–esa mission. In *2015 Conference of the International Fuzzy Systems Association and the European*

Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15). Atlantis Press, 2015.

[53] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[54] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[55] Giuseppe Vettigli. Minisom: minimalistic and numpy-based implementation of the self organizing map, 2018.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 147–153, 2003.

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[59] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Som toolbox for matlab 5. *Helsinki University of Technology, Finland*, 109, 2000.

[60] Teuvo Kohonen. Matlab implementations and applications of the self-organizing map. *Unigrafia Oy, Helsinki, Finland*, 2014.

[61] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[62] Usha A Kumar and Yuvnish Dhamija. Comparative analysis of som neural network with k-means clustering algorithm. In *2010 IEEE International Conference on Management of Innovation & Technology*, pages 55–59. IEEE, 2010.

[63] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[64] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[65] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[66] Nan Liu, Jinjun Wang, and Yihong Gong. Deep self-organizing map for visual classification. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–6. IEEE, 2015.

[67] Chathurika S Wickramasinghe, Kasun Amarasinghe, and Milos Manic. Deep self-organizing maps for unsupervised image classification. *IEEE Transactions on Industrial Informatics*, 15(11):5837–5845, 2019.

[68] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

[69] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

[70] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. *Advances in neural information processing systems*, 27, 2014.

[71] Laine Samuli and Aila Timo. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, volume 4, page 6, 2017.

[72] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[73] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[74] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[75] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[76] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[77] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[78] Alessandra Neri, Maria Camuglia, Alessia Tuninetti, Elena Baralis, Flavio Giobergia, and Davide Tricarico. Method and system for predicting system status, August 23 2022. US Patent 11,423,321.

[79] Mohammad Mesgarpour, Dario Landa-Silva, and Ian Dickinson. Overview of telematics-based prognostics and health management systems for commercial vehicles. In Jerzy Mikulski, editor, *Activities of Transport Telematics*, pages 123–130, Berlin, Heidelberg, 2013.

[80] Kihoon Choi, Satnam Singh, Anuradha Kodali, Krishna R Pattipati, John W Sheppard, Setu Madhavi Namburu, Shunsuke Chigusa, Danil V Prokhorov, and Liu Qiao. Novel classifier fusion approaches for fault diagnosis in automotive systems. *IEEE Transactions on Instrumentation and Measurement*, 58(3):602–611, 2008.

[81] Hong-Bae Jun, Dimitris Kiritsis, Mario Gambera, and Paul Xirouchakis. Predictive algorithm to determine the suitable time to change automotive engine oil. *Computers & Industrial Engineering*, 51(4):671–683, 2006.

[82] Uferah Shafi, Asad Safi, Ahmad Raza Shahid, Sheikh Ziauddin, and Muhammad Qaiser Saleem. Vehicle remote health monitoring and prognostic maintenance system. *Journal of advanced transportation*, 2018, 2018.

[83] Rune Prytz, Sławomir Nowaczyk, Thorsteinn Rögnvaldsson, and Stefan Byttner. Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering Applications of Artificial Intelligence*, 41:139–150, 2015.

[84] Danilo Giordano, Eliana Pastor, Flavio Giobergia, Tania Cerquitelli, Elena Baralis, Marco Mellia, Alessandra Neri, and Davide Tricarico. Dissecting a data-driven prognostic pipeline: A powertrain use case. *Expert Systems with Applications*, 180:115109, 2021.

[85] Thyago P. Carvalho, Fabrízzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, and Symone G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.

[86] Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. 2019. arXiv preprint. https://arxiv.org/abs/1912.07383.

[87] Samir Khan and Takehisa Yairi. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265, 2018.

[88] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: a survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019.

[89] Peter Wolf, Artur Mrowca, Tam Thanh Nguyen, Bernard Bäker, and Stephan Günnemann. Pre-ignition detection using deep neural networks: A step towards data-driven automotive diagnostics. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems*, 2018.

[90] Huan Luo, Miaohua Huang, and Zhou Zhou. A dual-tree complex wavelet enhanced convolutional lstm neural network for structural health monitoring of automotive suspension. *Measurement*, 137:14–27, 2019.

[91] Pooja Kamat and Rekha Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S Web of Conferences*, volume 170, page 02007. EDP Sciences, 2020.

[92] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *arXiv preprint arXiv:2206.09426*, 2022.

[93] Michael M Moser, Christopher H Onder, and Lino Guzzella. Using exhaust pressure pulsations to detect deteriorations of oxygen sensor dynamics. *Sensors and Actuators B: Chemical*, 191:384–395, 2014.

[94] Kübra Ekinci and Şeniz Ertuğrul. Model based diagnosis of oxygen sensors. In *Proocedings of the 9th IFAC Symposium on Advances in Automotive Control AAC*, 2019.

[95] Q Xin. Diesel aftertreatment integration and matching. In *Diesel Engine System Design*, pages 503–525. 12 2013.

[96] Teresa Donateo and Mattia Giovinazzi. Building a cycle for real driving emissions. *Energy Procedia*, 126:891–898, 2017.

[97] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[98] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.

[99] Haizhou Wang and Mingzhou Song. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal*, 3(2):29, 2011.

[100] Saúl Solorio-Fernández, J Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53(2):907–948, 2020.

[101] Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[102] Yijuan Lu, Ira Cohen, Xiang Sean Zhou, and Qi Tian. Feature selection using principal feature analysis. In *Proceedings of the 15th ACM international conference on Multimedia*, 2007.

[103] Jolliffe I.T. *Principal Component Analysis*. Springer-Verlag New York, 2002.

[104] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[105] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[106] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[107] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau. Random forests: some methodological insights. 2008. arXiv preprint. https://arxiv.org/abs/0811.3619.

[108] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *proceedings of the 31st international conference on distributed computing systems workshops*, 2011.

[109] Ramon Diaz-Uriarte and Sara Alvarez de Andrés. Variable selection from random forests: application to gene expression data. 2005. arXiv preprint. https://arxiv.org/abs/q-bio/0503025.

[110] Alain Rakotomamonjy. Variable selection using svm-based criteria. *Journal of machine learning research*, 3(Mar):1357–1370, 2003.

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

[112] Vitor Cerqueira, Nuno Moniz, and Carlos Soares. Vest: Automatic feature engineering for forecasting. *Machine Learning*, pages 1–23, 2021.

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

[114] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[115] Flavio Giobergia, Elena Baralis, Maria Camuglia, Tania Cerquitelli, Marco Mellia, Alessandra Neri, Davide Tricarico, and Alessia Tuninetti. Mining sensor data for predictive maintenance in the automotive industry. In *Proceedings of the 5th International Conference on Data Science and Advanced Analytics*, 2018.

[116] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[117] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[118] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[119] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[120] Danilo Giordano et al. PREPIPE, 2021. https://github.com/SmartData-Polito/PREPIPE.

[121] Pabitra Mitra. *Pabitra Mitra personal webpage*, (accessed June 4th, 2020). http://cse.iitkgp.ac.in/~pabitra/paper/fsfs.tar.gz.

[122] Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE transactions on neural networks*, 14(2):274–281, 2003.

[123] D Stathakis. How many hidden layers and nodes? *International Journal of Remote Sensing*, 30(8):2133–2147, 2009.

[124] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[125] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[126] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE conference on computer vision and pattern recognition*, 2012.

[127] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[128] Biao Wang, Yaguo Lei, Tao Yan, Naipeng Li, and Liang Guo. Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery. *Neurocomputing*, 379:117–129, 2020.

[129] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[130] Jiujian Wang, Guilin Wen, Shaopu Yang, and Yongqiang Liu. Remaining useful life estimation in prognostics using deep bidirectional lstm neural network. In *Proceedings of Prognostics and System Health Management Conference*, 2018.

[131] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[132] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[133] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[134] Christopher Moseley. *Atlas of the World's Languages in Danger*. Unesco, 2010.

[135] Alan Ramponi. Nlp for language varieties of italy: Challenges and the path forward. *arXiv preprint arXiv:2209.09757*, 2022.

[136] Dongrui Wu. Pool-based sequential active learning for regression. *IEEE transactions on neural networks and learning systems*, 30(5):1348–1359, 2018.