Journal of Intelligent Information Systems
https://doi.org/10.1007/s10844-024-00841-4

RESEARCH

Data- & compute-efficient deviance mining via active learning and fast ensembles

Francesco Folino[1] · Gianluigi Folino[1] · Massimo Guarascio[1] · Luigi Pontieri[1]

Received: 28 July 2023 / Revised: 1 November 2023 / Accepted: 10 January 2024

Abstract
Detecting deviant traces in business process logs is crucial for modern organizations, given the harmful impact of deviant behaviours (e.g., attacks or faults). However, training a Deviance Prediction Model (DPM) by solely using supervised learning methods is impractical in scenarios where only few examples are labelled. To address this challenge, we propose an Active-Learning-based approach that leverages multiple DPMs and a temporal ensembling method that can train and merge them in a few training epochs. Our method needs expert supervision only for a few unlabelled traces exhibiting high prediction uncertainty. Tests on real data (of either complete or ongoing process instances) confirm the effectiveness of the proposed approach.

Keywords Process deviance · Deep ensembles · Active learning · Green AI · XAI · Log analysis

1 Introduction

(Process) *Deviance mining* (Folino & Pontieri, 2019) refers to the problem of detecting and analyzing traces exhibiting "deviant" behaviors in a process log. This problem received increasing attention recently, as deviant executions (e.g., linked to faults, attacks or frauds) can impact seriously on business process quality. For example, deviance mining techniques

Francesco Folino, Gianluigi Folino, Massimo Guarascio, and Luigi Pontieri contributed equally to this work.

✉ Luigi Pontieri
luigi.pontieri@icar.cnr.it

Francesco Folino
francesco.folino@icar.cnr.it

Gianluigi Folino
gianluigi.folino@icar.cnr.it

Massimo Guarascio
massimo.guarascio@icar.cnr.it

1 ICAR, CNR, Via P. Bucci 8-9C, Rende 87036, CS, Italy

can be used to discover malicious behaviours in container logs, linked to hidden malware and/or covert attacks.

Most existing solutions (e.g., (Bose & van der Aalst, 2013; Nguyen et al., 2014; Lo et al., 2009; Cuzzocrea et al., 2016b; Folino et al., 2013)) reuse Machine Learning (ML) methods to train a *Deviance Prediction Model* (*DPM*), i.e., a classifier discriminating deviant traces from normal ones, on propositional trace representations derived by extracting relevant behavioural patterns from the traces and using them as data features. Notably, such a pattern-based approach allows for dealing with interpretable data representations, which will turn helpful when trying to explain/justify the predictions that the discovered DPM makes on new traces. In particular, Bose & van der Aalst (2013); Nguyen et al. (2014) showed the advantage of mixing up different pattern families for the sake of higher expressivity, while Cuzzocrea et al. (2016b) proposed to train an ensemble of DPMs via a multi-view learning strategy, where each DPM is induced from a distinct pattern-based view of the training data.

Starting from empirical evidence of the superiority of deep models w.r.t. shallow ML ones in predictive process mining tasks, the approach in Cuzzocrea et al. (2016b) was recently enhanced in Folino et al. (2020) by using *Deep Neural Networks* (*DNNs*) to implement the base DPMs of the ensemble and different combination functions. Notably, instead of training an NN model with a sequence-oriented architecture (like Recurrent and Transformer-based NNs), for the sake of explainability, efficiency and generalization, also Folino et al. (2020) keeps on using a (pattern-based) flat representation of the traces to train a simpler MLP-like classifier.

Small training data problem, active learning for DPM discovery and open issues As noted in Cuzzocrea et al. (2016b), in many real-life deviance mining scenarios, a small fraction of log traces have a known deviance-class label, as labelling a process instance as either deviant or normal is time-consuming and costly. Notably, such scenarios can be addressed effectively and efficiently by leveraging an *Active Learning* (*AL*) approach (Ren et al., 2021), where human experts are put in the learning loop by repeatedly asking them to inspect and label few unlabelled log traces (chosen based on prediction uncertainty) to help improve the current DPM.

Example 1 (running example) Consider a hospital process for handling gynecologic patients, like that originating the dataset $\mathtt{BPI}_{dM16}$ used in the tests of Section 6. Detecting patients suffering from relatively rare cancer-related diseases can be regarded as a special kind of supervised deviance mining task, which can be addressed by leveraging a powerful (deep) DPM trained on labelled patient traces. However, few labelled patient traces are usually available in such a scenario, so that the resulting DPM may well deem healthy individuals as at-risk and overlook critical cases. Having skilled physicians manually label large amounts of patient traces is impractical because of the limited availability and high costs of such specialized staff. More efficiently, one can exploit an human-in-the-loop AL-based training approach, where only few unlabeled cases are passed to the expert, in order to be labelled. Whenever re-trained over the collection of examples augmented this way, the DPM is expected to improve progressively, and eventually reach satisfactory accuracy performances.

However, adapting AL strategies to discover deep DPMs poses some critical issues. First, as the class-membership probabilities returned by such classifiers are not calibrated, selecting the examples to label based on (uncertainty scores derived from) these probabilities may perform worse than random sampling (Ren et al., 2021). On the other hand, though reasonable uncertainty estimates can be obtained with Bayesian DNNs or DNN ensembles, such models are far more costly to train and apply (Ren et al., 2021), and less suitable for interactive AL settings.

Anyway, deep models need to be trained with large amounts of (labelled or unlabelled) example data, which may not be available in several real-life contexts, especially in Green-AI applications where restrictions to data accesses and data processing operations hold (Adadi, 2021). Thus, the DNN architecture and training method must be devised carefully to reduce the risk of obtaining an under/over-fitted DPM when using a relatively small amount of training data.

Contribution In light of the considerations above, a novel approach to the DPM discovery problem is addressed here, in both offline and online settings (i.e. for completed and running process instances, respectively), which leverages a synergistic combination of ad hoc (data efficient) Active Learning and (Fast) Ensemble Learning schemes, in order to compensate for the scarcity and approximated representation of the training sample by exploiting a carefully devised discovery approach.

This paper extends the work presented in Folino et al. (2022) to a substantial extent. Some major technical features of our proposal are:

1. All the base DPMs are devised to ingest propositional trace representations obtained with interpretable and compact trace encoding schemes (see Section 2), including two of those used in Cuzzocrea et al. (2016b); Folino et al. (2020); Bose & van der Aalst (2013); Nguyen et al. (2014) to curb the feature sparsity and heterogeneity and the computational costs.
2. For the sake of efficiency and feasibility in an AL-oriented setting, we discover an approximated DPM ensemble via a fast temporal ensemble-learning strategy (Huang et al., 2017) (see Section 5) that finds diverse deep DPMs in as many training epochs as those needed to optimize a single DNN of the same structure and use non-trainable functions to fuse the base DPMs' predictions; to ensure repeated rapid convergence to good local optima, we employ a cyclic learning-rate schedule and a DNN architecture with both dropout and residual-like components (see Section 5.2). Then, the unlabeled instances to pass to the expert are chosen according to their prediction uncertainty, estimated based on how the base DPMs probabilistically classified them (see Section 5).
3. In this work, we have also explored the usage of *Soup* ensembles (based on fusing multiple snapshot-like instances of a DNN in the parameter space, rather than on combining their output) (Neyshabur et al., 2020). Applying such a model takes the same costs, in terms of both memory and FLOPs, as a single DNN instance, while training it requires less memory than training a snapshot ensemble like those discovered in Folino et al. (2022). As confirmed by test evidence, the resulting condensed DPM manages to achieve nearly the same prediction performances as the DPM ensemble it was derived from, while reducing the computation cost of inference steps of a factor equal to the number of snapshots (i.e., of base DPMs).

Experiments conducted on real-life log data (Section 6) showed that this approach yields compelling accuracy results at the cost of reasonable efforts by the experts, even if compared to the results that (computationally more costly) state-of-the-art methods obtained in the ideal unlikely scenario where all log traces are labelled. This confirms its ability to effectively deal with a challenging mix of data/energy efficiency, accuracy and explainability requirements that will likely arise in the near future across many application contexts. The potential usefulness of the proposed approach in the healthcare scenario of our running example is provided below.

Example 2 (contd.) Assume that the AL-based framework proposed here is deployed in the deviance mining scenario of Example 1 having some expert physicians to revise the class labels of 20 process instances (selected as the unlabelled ones getting the 20 most uncertain predictions from the DPM ensemble) every two days along a 6-day period. This operational setting was simulated in the tests that we performed on dataset $\mathtt{BPI}_{dM16}$ to validate the proposed approach (see Section 6.3). Interestingly, the accuracy achievements obtained in this tests were quite satisfactory, at the cost of quite a light burden for the experts (a total of 60 instances to label in 6 days) and relatively small computational cost in terms of both time (3 training sessions of just 32 epochs each) and memory (when using the Soup ensemble mode, the size of the entire ensemble is the same as those of one base model). Note that the labeling task performed by the expert can be eased and sped up by providing her with easy-to-interpret post-hoc explanations (like those reported in Section 6.5) for the predictions that the DPM ensemble suggests for the instances she is asked to label.

Organization The rest of the paper is structured as follows. Section 2 introduces a number of basic concepts (including trace encoding and scalable deep ensembles). The problem of discovering a DPM is stated in Section 3, for both online and offline deviance mining settings. Section 4 offers an overview of major ML/DL approaches to deviance mining. Section 5 illustrates our AL-based approach to induce an ensemble of deep DPMs. Experiments on real-life log data are discussed in Section 6, while concluding remark and feature research directions are drawn in Section 7.

2 Background

Process events, traces and logs As usually done in Deviance Mining, we assume that, for every execution instance of the process under analysis (a.k.a. *process instance*), a distinguished *trace* is stored, which consists of a (temporally ordered) sequence of *events*, plus several instance-level attributes that do not vary along a process execution. Each event is a tuple storing the activity that was executed correspondingly to the event and possibly more information (e.g., the executor, data parameters). At run time, during the unfolding of a process instance, a partial trace is recorded for it, storing the sequence of processing events stored so far for the process instance. Such a trace is called a *prefix trace* and represents a sort of pre-mortem log data.

Essentially, in this work, we want to devise a process mining approach to the discovery of a *Deviance Prediction Model* (*DPM*), which can support the analyst in deciding on the deviant nature of a process instance, based on its associated trace –be it the complete trace of a fully executed instance (offline deviance prediction) or the prefix trace (online deviance prediction) of a running instance. This problem was often addressed in the literature by employing propositional log encoding instead of directly dealing with event sequences, in order to reuse standard ML solutions, or for the sake of efficiency and explainability. Some of the proposed encodings are described below, before defining the problem setting more precisely.

Propositional trace encodings In many previous approaches to the discovery of a DPM (Bose & van der Aalst, 2013; Nguyen et al., 2014; Lo et al., 2009; Cuzzocrea et al., 2016b), all focusing on the offline setting, e traces are turned into symbolic sequences (usually by abstracting each event into its associated activity label), before extracting a number of sequential patterns from these symbolic sequences and then using them as features to build up a fixed-length encoding of the traces. Different kinds of patterns have been used to this end so far: *individual activity* patterns, *discriminative pattern*, and several pattern types borrowed from bio-informatics (e.g., tandem/maximal repeats and their alphabet-abstracted versions).

Specifically, *Individual activity* (**IA**) (Suriadi et al., 2013) patterns just correspond to regarding every activity label a as a distinct feature (sort of "singleton" execution pattern) for any trace τ, and use this feature to store how many times a occurs in τ. *Discriminative patterns* (**DP**) (Lo et al., 2009) are frequent, closed, possibly non-contiguous sub-sequences of activity labels aimed at capturing behaviours that repeatedly occur in either a trace or multiple ones. DP patterns can be found efficiently using the CLIPPER algorithm of Lo et al. (2009), which computes frequent candidates via an a-priori-like computation and then returns the most discriminative ones based on the Fischer score. Each DP pattern is eventually used as a non-negative integer feature for any trace τ, storing how frequently the pattern occurs in τ.

Multi-perspective trace encoding methods have also been proposed in the literature, which take into account other event attributes than the activity-related one, According to the empirical study of Teinemaa et al. (2019) and subsequent ones, the so-called *aggregation encoding* (**AE**) looks the best performing one among these.

This encoding generalizes the IA-based one by applying specific aggregation functions to every event attribute, all over the trace steps, namely, the AVG and STD functions to numerical attributes (e.g., duration) and the COUNT function to categorical ones (including activity labels, as done for IA).

All the above-described encoding schemes can be easily extended to incorporate global instance properties (case attributes).

Fast (Snapshot/SOUP) DNN ensembles An efficient method for learning an ensemble of predictive DNN models (e.g., classifiers), named *snapshot ensemble*, was introduced in Huang et al. (2017). The key idea behind this method is that using a cyclic learning rate schedule, a model explores different regions of the loss landscape during training, potentially finding different local optima in each cycle. Formally, given a training set D enriched with the ground-truth-labeled instances in X (and a validation set D_{VAL}), the DNN can be trained in the required number e of epochs by making it converge to k local minima along its optimization path: the model parameters obtained for each minimum are saved in the ensemble as a separate base model of the ensemble. A cyclic learning rate schedule is combined with a classic SGD scheme to ensure repeated rapid convergence.

The idea behind the *SOUP* ensembles draws inspiration from the insights presented in Neyshabur et al. (2020). This research demonstrated that when multiple models are independently fine-tuned based on an identical base model, they inherently operate within the same loss landscape basin. Based on this premise, interpolating two solutions (i.e., linearly combining their respective parameter weights) could conceivably position the result nearer to the basin's epicentre. The main advantage of a SOUP model is its ability to create a single-model (robust and reliable) classifier by merging multiple models in the space of parameters. This avoids the need to use additional memory or computational time during inference. The simplest way to build such a SOUP is to compute the parameter weights of the resulting model as the average of the corresponding ones in the original models of the ensemble. This approach is known as *uniform soup*. Though more sophisticated merging techniques were proposed, such as *greedy soups* and *learned soup* (Wortsman et al., 2022), we opt for the uniform approach for the sake of efficiency.

3 Problem statement: offline and online deviance mining

We here want to state the problem of discovering a DPMs from a collection of log traces, after turning them into a propositional form. Given a training set D of propositionally-encoded log traces, we assume that only the training instances in a subset $D^L \subset D$ are equipped with a known class label in $\{deviant, normal\}$, while the remaining subset $D^U = D \setminus D^L$ consists of unlabeled instances.

As a preliminary step for the discovery/application of a DPM, each log trace (be it complete or a prefix one) τ is assumed to have been turned into a fixed-length tuple $\overrightarrow{\tau} = x_1, \ldots, x_N$, where each x_i is either one the features that result from applying a trace encoding scheme (e.g., one of the IA, DP and AE -based ones in Section 2) to τ or one of the constant case attributes (encoded in one-hot, if categorical) in the τ. Denoting as $X_1, \ldots, X_N$ the domains these features are defined on, we will refer to the universe of such data instances as $\mathscr{U} = X_1 \times X_2 \times \ldots \times X_N$. Several such encoded-trace instances are assumed to be available, logically partitioned into two subsets: a set D^L of instances equipped with a deviance-class label in $\{normal, deviant\}$ and a set D^U of unlabelled instances.

The problem of learning a DPM (for offline/online analyses) is stated below.

Definition 1 (Offline and online DPM discovery problem) Let $D = D^L \cup D^U$ be a given sample of data instances representing different complete traces (resp., prefix traces) of a process, such that D^L and D^U gather the labelled and unlabelled, respectively, instances in D. Then, the *offline* (resp., online) *deviance mining* problem amounts to training, using D^L and D^U, a *Deviance Prediction Model* (DPM) that can estimate, for any novel (feature-encoded) trace τ, the probability that the process instance generating τ was (resp., will eventually be) deviant.

As usual, the labelled example traces in D^L can be provided as input to any existing supervised deviance mining approach in the literature in order to extract a DPM. In order to also exploit the data in D^U and human expert's supervision, we propose to approach the problems above by using an Active Learning strategy, where a DPM trained on D^L is improved incrementally, by using additional labelled examples, obtained by having the expert label few (wisely chosen) instances from D^U. Specifically, we assume that two parameters characterize the effort that the human expert is willing to provide over the time: the number m of active learning steps, and the number b of instances that (s)he can manually label in each of these steps. We will call b and $b_T = b \times m$ the active learning *budget* and *total budget*, respectively.

4 Related work

Deviance mining in business processes has been commonly addressed in an offline (post-mortem) prediction setting, where the traces to be classified (as either deviant or normal) are all complete ones. Different solutions have been proposed in the literature, which can be categorized into three groups: supervised, unsupervised, and hybrid deviance mining approaches. For the sake of space, this section focuses on supervised approaches, which are closest to our work, in that they also employ a supervised learning method to discover a classification model for discriminating among "normal" and "deviant" traces, from a given collection of labelled traces. The discovered model can be used to decide whether a novel trace is deviant or not (*deviance detection*), while possibly trying to understand which trace features help explain why a deviance has occurred (*deviance explanation*).

Supervised deviance mining (for offline deviance analyses) A selection of supervised approaches to deviance mining is reported in Table 1. A typical approach in classification-based deviance mining employs a two-phase scheme: (1) a propositional encoding scheme is employed to convert the traces into a vectorial form; (2) a classifier is induced from labelled vectors of this form (where the label represents the deviance class), by using some machine learning method. As mentioned in Section 2, some of the proposed encoding schemes rely on extracting behavioural patterns from the given traces and use these patterns as distinguished data features.

For instance, Suriadi et al. (2013) utilized an IA-based log encoding to train a decision tree to discriminate and explain deviances defined as traces with excessively long durations. Swinnen et al. (2012) proposed a semi-automatic deviation analysis method that mixes process mining and association rule mining methods. Bose & van der Aalst (2013) concentrated on the discovery of a binary classifier from traces encoded on the basis of alternative pattern families (e.g., IA, TR, MR, SMR, NSMR, etc.), with deviances capturing fraudulent or faulty cases. Focusing on the case of software trace failures, Lo et al. (2009) introduced the concept of *closed unique iterative patterns* (IPs) as encoding features –from which one can extract a subset of discriminative ones, named Discriminative Patterns (DP) in Nguyen et al. (2014). Nguyen et al. (2014) proposed a comprehensive scheme unifying previous pattern-based deviance mining works, which includes oversampling, pattern mining, pattern selection, pattern-based log encoding, and classifier induction.

In addition to these, there are also multi-encoding hybrid methods that combine different kinds of patterns and classifier-induction algorithms. These methods, such as the ensemble-based one proposed by Cuzzocrea et al. (2015, 2016b), automatically discover and combine different patterns and induction algorithms to create a robust model for deviance prediction. Cuzzocrea et al. (2016a) combined a simplified version of this approach with a conceptual event clustering method to extract payload-aware event abstractions automatically.

Table 1 Major supervised deviance mining approaches proposed in the literature

Task	Strategy	Reference
Detection, Explanation	Classifier induction (IA patterns)	Suriadi et al. (2013)
Detection, Explanation	Classifier induction (SET patterns)	Swinnen et al. (2012)
Detection, Explanation	Classifier induction (non IP sequence patterns)	Bose & van der Aalst (2013)
Detection, Explanation	Classifier induction (iterative patterns (IPs))	Lo et al. (2009)
Detection, Explanation	Classifier induction (IA and sequence patterns)	Nguyen et al. (2014)
Detection	Classifier ensemble induction (IA and sequence patterns)	Cuzzocrea et al. (2015)
Detection	Classifier ensemble induction (IA and sequence patterns)	Cuzzocrea et al. (2016b)
Detection	Classifier ensemble induction (“abstract” IA and sequence patterns)	Cuzzocrea et al. (2016a)
Explanation	Subgroup Discovery	Fani Sani et al. (2017)
Explanation	Deviance-oriented Conceptual Clustering	Folino et al. (2017)

Other approaches within supervised deviance mining include the use of subgroup discovery techniques (Atzmueller, 2015) and deviance-aware conceptual clustering. In particular, Fani Sani et al. (2017) proposed to extract explanations from labelled traces by applying classic subgroup discovery techniques. This method identifies unusual trace subgroups regarding class distribution, providing valuable insights into deviant behaviours. Folino et al. (2017) adopted a different approach, focusing on a setting where intriguing self-explaining deviance scenarios are to be extracted from training traces associated with numerical deviance indicators.

Online deviance detection and predictive process monitoring To the best of our knowledge, the problems of estimating whether a running process instance is deviant or not (or the probability of it being deviant) has never been addressed explicitly as a predictive learning task in the literature. In fact, in online settings, the anticipated prediction (forecast) of deviant behaviours has been commonly faced via reasoning-based approaches, e.g., in the contexts of security breach detection (Fazzinga et al., 2018) and of compliance monitoring (Ly et al., 2015).

However, as noted by Rinderle-Ma & Winter (2022), forecasting whether a running process instance will deviate from a compliance model/constraint has sometimes been faced as a form of predicate/outcome prediction in the field of *Predictive Process Mining* (Di Francescomarino & Ghidini, 2022). In this field, several deep learning approaches were proposed of late, based on feed-forward NNs and more expressive sequence-oriented DL (e.g., recurrent, convolutional or transformer-based) architectures (Neu et al., 2022).

Compelling accuracy results were also obtained with multi-view learning schemes (Pasquadibisceglie et al., 2021a; Cuzzocrea et al., 2016a). However, the higher expressivity of such DL models, compared to feed-forward neural networks applied to pattern-based trace encodings are likely to pose more difficulty in obtaining reliable and understandable explanations for the returned predictions.

5 The proposed AL-based framework

Our approach learns a (homogenous) ensemble of DNNs, all sharing the architecture presented in Section 5.2, which takes the form defined below.

Definition 2 (DPM Ensemble) A *DPM Ensemble* $\mathscr{M}$ over the universe $\mathscr{U}$ of data instances (trace encodings) is a tuple of the form $\langle M_1, \ldots, M_k, \phi \rangle$ for some $k \in \mathbb{N} \setminus \{1\}$, such that: **(i)** for each $i \in [1..k]$, $M_i : \mathscr{U} \to [0, 1]$ is a DNN model that maps any instance $\overrightarrow{\tau} \in \mathscr{U}$ to a "deviance score" $M_i(\overrightarrow{\tau})$ estimating the probability that the $\overrightarrow{\tau}$ corresponds to a deviant process instance, and **(ii)** $\phi : [0, 1]^k \to [0, 1]$ is an aggregation operator that, for any $\overrightarrow{\tau} \in \mathscr{U}$, merges the predictions returned by $M_1, \ldots, M_k$ for $\overrightarrow{\tau}$ into an overall one $\phi(\{M_1(\overrightarrow{\tau}), \ldots, M_k(\overrightarrow{\tau})\})$. Hereinafter, we will call $M_1, \ldots, M_k$ the *base models* of $\mathscr{M}$ and ϕ the *combiner* of $\mathscr{M}$.

DPM variants: ensembling modes In our approach, five different *ensembling modes* can be used to instantiate the structure and behaviour of the DPM ensemble, and give rise to five different variants this (abstract) class of prediction models.

Three of these modes apply to the standard case where the ensemble actually includes multiple base models and allow for choosing among three different aggregation functions:

the maximum (mode MAX), average (mode AVG) and median (mode MEDIAN). Notably, these fixed combination functions are a faster (and more suitable for green-aware and Active Learning settings) alternative to the trainable combiners of Cuzzocrea et al. (2016b); Folino et al. (2020).

A further ensembling mode, SOUP, is devised to build a single-DNN DPM (i.e., a DPM ensemble consisting of one model) by merging multiple base DPMs in the space of model parameters (see Section 2). This allows for having just one DNN (of the same form as the base DPMs) enjoying ensemble-like prediction abilities.

The last ensembling mode, NONE, serves the purpose of training a single base DPM, using no ensemble learning strategy at all. Clearly, when using ensembling modes SOUP and NONE, the role of the aggregation function gets immaterial, for there are not multiple base models to combine.

5.1 Learning algorithm

Algorithm 1 illustrates, in the form of pseudo-code, our *Active Learning* (*AL*) approach to the discovery of a DPM ensemble of the forms introduced in the previous subsection. The algorithm assumes that a human expert (or a group of experts) is available to provide supervision for updating the current DPM ensemble, by labeling a batch b of unlabeled data instances for a number m of times. If wanting to deal with a stream-like D^U, one can set $m = \infty$ and use the current version of $\mathscr{M}$ at any moment without waiting for the end of the algorithm.

A first version of the DPM ensemble is built by looking only at the given set in D^L of labelled data instances by leveraging a fast ("snapshot-based") approximated ensemble-learning procedure, denoted in the algorithm as train_ensemble. To improve this DPM ensemble, limited feedback is acquired from experts, who are asked to iteratively analyze and label a small number (controlled by parameter b, named from now on the *expert budget*) of instances selected from the given set D^U of unlabelled instances. The choice is based on prediction uncertainty scores (higher scores are preferred), in order to provide the learning

Algorithm 1 Lifecycle of a DPM ensemble model in our approach.

```
1: Requires: labelled data $D^L \subseteq \mathscr{U}$, unlabelled data $D^U \subseteq \mathscr{U}$, a human expert available to provide supervision in $m \in \mathbb{N} \cup \{\infty\}$ AL steps by labeling $b \in \mathbb{N}$ instances per step, ensembling mode $mode \in \{$MAX, AVG, MEDIAN, SOUP, NONE$\}$, number $k$ of base models (with $k = 1$ if $mode \in \{$SOUP, NONE$\}$), number $e \in \mathbb{N}$ of training epochs, initial learning rate $lr$, validation percentage $val\_perc$;
2: Returns: a DPM ensemble $\mathscr{M} = \langle M_1, \ldots, M_k, \phi \rangle$ conforming to the ensembling mode specified through parameter $mode$.
3: Split $D^L$ into training/validation sets $D^L_{TR}$ and $D^L_{VAL}$ s.t. $|D^L_{VAL}| = |D^L| \times val\_perc/100$;
4: $\mathscr{M}$ := train_ensemble($D^L_{TR}, D^L_{VAL}, k, mode, epochs$);
5: for $i = 1..m$ do
6:     wait for condition ($|D^U| \geq b$ and an expert is available for labelling) = true;
7:     Choose subset $X \subseteq D^U$ of $b$ tuples $\mathscr{M}$ was most uncertain on (based on (2) or (1))
8:     $D^L_{TR} = D^L_{TR} \cup X$;
9:     $\mathscr{M}'$ := train_ensemble($D^L_{TR}, D^L_{VAL}, k, mode, epochs$);
10:    if $\mathscr{M}'$ performs better than $\mathscr{M}$ over $D^L_{VAL}$ then    // according to F1 scores
11:        $\mathscr{M} := \mathscr{M}'$;
12:    end if
13: end for
```

procedure with novel labelled examples as informative as possible —details on this respect are given later on.

Procedure `train_ensemble` Abstracting from the trivial case of training just one base model (ensembling mode NONE), this procedure follows the fast *snapshot ensemble* approach in Huang et al. (2017) (see Section 2). Given the set D^L_{TR} of labelled instances and a validation set D^L_{VAL}, we train a DNN of the form described in Section 5.2 in the required number e of epochs, by making it converge to k local minima, named *snapshots*, along its optimization path, and regarding the parameter weights found at each snapshot as (the configuration of) a distinct base model. Precisely, the first base model M_1 of the ensemble is randomly initialized and trained for $\lfloor e/k \rfloor$ epochs with a variable learning rate η: η is initially set to lr and then progressively lowered, from one epoch to the next, according to a shifted-cosine schedule (Huang et al., 2017).

In particular, when using one of the ensembling modes MAX, AVG and MEDIAN, k different *snapshot models* are computed and kept as different base models: for $1 < i \leq k$ a novel base model M_i is initialized with the same weights as M_{i-1}, and trained by setting η again to lr, and eventually selecting for it the parameter configuration achieving the highest F1 score on D^L_{VAL} over the different training epochs.

A variant of this procedure is used for mode SOUP, as there is no need to store all the snapshot models but only to compute the average of their respective parameter weights. Thus, the final ensemble-like model is computed by averaging the snapshot-wise parameter weights incrementally, using just the parameter weights at the current snapshot and the sum/average of the parameter weights of all the previous snapshots.

Uncertainty estimation Let $\mathscr{M} = \langle M_1, \ldots, M_k, \phi \rangle$ be a DPM ensemble and $x \in \mathscr{U}$ be any data instance representing some trace τ. Let $p^q(x) = M_q(x)$ be the prediction returned, for x by the q-th model in the ensemble, and $p^{\mathscr{M}}(x) = \phi(\{M_i(x) \mid i \in [1..k]\})$ be the prediction returned for x by the ensemble as a whole.

The total uncertainty affecting the ensemble prediction can be quantified as the entropy score $H^{\mathscr{M}}_{tot}(x) = -p^{\mathscr{M}}(x) \times \log p^{\mathscr{M}}(x) - \left(1 - p^{\mathscr{M}}(x)\right) \times \log\left(1 - p^{\mathscr{M}}(x)\right)$.
This value sums up two components: an *aleatoric* (or irreducible) component $H^{\mathscr{M}}_{al}(x)$ and an *epistemic* one $H^{\mathscr{M}}_{ep}(x)$, the latter of which could be possibly reduced by using further training examples in the learning process.

The aleatoric uncertainty can be estimated as $H^{\mathscr{M}}_{al}(x) \approx \frac{1}{k}\sum_{q=1}^{k} H^q(x)$, where H^q is the entropy related to the prediction of each base model M_q, i.e.,

$$H^q(x) = -p^{(q)}(x) \times \log p^{(q)}(x) - \left(1 - p^{(q)}(x)\right) \times \log\left(1 - p^{(q)}(x)\right) \tag{1}$$

The epistemic uncertainty involved in using $\mathscr{M}$ to classify x is quantified as the difference between the total and aleatoric entropies $H^{\mathscr{M}}_{ep} \approx H^{\mathscr{M}}_{tot} - H^{\mathscr{M}}_{al}$, namely:

$$H^{\mathscr{M}}_{ep} \approx -p^{\mathscr{M}}(x) \times \log p^{\mathscr{M}}(x) - \left(1 - p^{\mathscr{M}}(x)\right) \times \log\left(1 - p^{\mathscr{M}}(x)\right) - \frac{1}{k}\sum_{q=1}^{k} H^q(x) \tag{2}$$

Algorithm 1 uses the uncertainty score of (2) to choose the unlabelled data to give to the expert when the DPM ensemble that is being built really consists of more than one base models (i.e., $mode \in$ {SOUP, NONE}). This strategy resembles the BALD criterion (Ren et al., 2021), which prefers unlabelled instances that get more diverging predictions; though the models in our ensembles are not trained truly independently, this approach is shown effective in practice in our tests (Section 6).

This equation cannot be used on singleton DPM ensembles (like those produced with the SOUP ensembling mode). In this case, we just compute the total entropy of the sole model in the ensemble, say M_q, by using (1). As to SOUP DPMs, we conjecture that this simple approximate estimate somewhat captures the uncertainty in the set of DNNs it was derived from. Anyway, we avoid using refined uncertainty estimation methods like Bayesian dropout, to keep our SOUP-based DPM ensembles as fast as possible (and suitable for online deviance detection).

5.2 Base DNN architecture and loss function

The base classifiers in the proposed ensemble-based DPM model share the same architecture, depicted on the left-hand side of Fig. 1. The architecture, including dropout and residual-like modules, enables rapid convergence and high robustness to over/under-fitting a few training examples.

In more detail, the architecture consists of the following stack of components: (i) an *input* layer furnishing the propositional pattern-based encoding of any input trace; (ii) three instances of a *Residual Block* sub-net (denoted in the figure as $RB_1, \ldots, RB_3$), which consist each of two instances of a *Building-block* sub-net linked one another by a skip connection as shown in the right-hand side of Fig. 1; (iii) an output layer consisting of a single neuron equipped with a *sigmoid* activation function, which eventually returns a deviance score in [0,1]. Each building block is composed in its turn of three components: *(i)* a fully-connected layer including 128 neurons with *tanh* activation functions, *(ii)* a batch-normalization layer and *(iii)* a dropout layer with a dropout rate of 0.25.

Loss function A weighted variant of the *Mean Absolute Error* (MAE) is used as the loss function for training the base DPM architecture, in order to deal with the case of unbalanced classes that frequently occurs deviance mining settings. The loss, which pays more attention to the (rarer) training instances of the minority deviant class, is defined as follows: $loss(y^{(i)}, \tilde{y}^{(i)}) = \frac{1}{n}\sum_{i=1}^{n} \|y^{(i)} - \tilde{y}^{(i)}\| \cdot weight(y^{(i)})$, where n is the number of instances in the training set, $y^{(i)}$ and $\tilde{y}^{(i)}$ are the real and predicted deviance score of the i-th instance, and $weight(y^{(i)})$ is the misclassification cost associated with the predicted class (i.e., the deviant class if $\tilde{y}^{(i)} > 0.5$ or the normal one otherwise). By default, the costs are set to 1 and 2 for the normal and deviant classes, respectively.

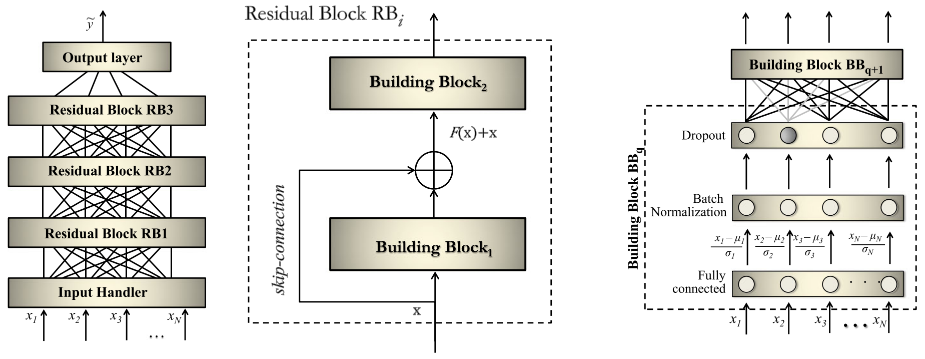


Fig. 1 The DNN Architecture adopted for all base DPM models in our approach (left), and details on the Residual Block (middle) and Building Block (right) sub-nets

6 Experimental evaluation

In this section, we test the capability of our ensemble-based strategies, which leverage an active learning (AL) strategy, to enhance their effectiveness in identifying deviant behaviours. This examination is performed in two distinct experimental scenarios: (1) an offline setting focused on the analysis of deviances in post-mortem log data; and (2) an online (early prediction) setting wherein we attempt to forecast the deviance of a trace during its unfolding. In both scenarios, we evaluate our solutions against a non-ensemble method and various ensemble strategies prevalent in the current literature. This comparative analysis is designed to highlight the benefits of our AL-centric approach within the context of real-world applications.

6.1 Datasets

Logs used for offline deviance mining tests As done in previous studies (Cuzzocrea et al., 2016b; Nguyen et al., 2014), we used a real-life log from a Dutch hospital (van Dongen, 2011), consisting of 1,142 traces, 150,291 events and 624 activities. Each trace stores the activities (treatments/tests) performed on gynecology patients and several case/event attributes. Case attributes include `Diagnosis`, `Diagnosis code`, `Treatment code` and `Age`, while event attributes include `Activity code`, `Specialism code`, and `Group`. Two derived case attributes (i.e. the case duration and number of trace events) were added to each trace.

Following the same classification criteria as in the studies above, we generated two datasets, dubbed $\texttt{BPI}_{dM13}$ and $\texttt{BPI}_{dM16}$. In these datasets, each trace was given a label, either *deviant* (label = 1) if the diagnosis code was ‘M13’ or ‘M16’, respectively, or ’normal’ (label = 0) otherwise. Summary statistics of these datasets are shown in Table 2. Notably, both datasets are class-imbalanced: 310 deviant traces vs 832 normal ones in $\texttt{BPI}_{dM13}$, and 216 deviant traces vs 926 normal ones in $\texttt{BPI}_{dM16}$.

In the traces of both datasets, we removed attributes (namely, `Diagnosis`, `Diagnosis code`, `Treatment code`) that may disclose class label information and eventually converted each trace into a tuple by concatenating the features produced for the former by the AI and DP encodings (see Section 2).

Logs for online deviance mining The dataset (Mannhardt, 2016) used in this setting registers a record of sepsis cases from a Dutch hospital from 2013-2015. The log stores patients’ journey from their first interaction in the emergency room to their eventual hospital discharge. Each trace in the dataset encodes the medical history of a patient undergoing various medical procedures during their hospital stay. For the sake of privacy, the information is anonymized. The dataset stores information on clinical procedures, diagnostic

Table 2 Summary statistics of used datasets

Dataset	Setting	#traces	#deviant	#regular	Avg. len.	Imbalanced
$\texttt{BPI}_{dM13}$	*offline*	1,142	310	832	109	Yes
$\texttt{BPI}_{dM16}$	*offline*	1,142	216	926	120	Yes
`sepsis_cases_2`	*online*	782	109	673	13	Yes
`sepsis_cases_3`	*online*	782	109	673	13	Yes

tests (e.g., `DiagnosticBlood`, `DiagnosticECG`) and associated results, as well as demographic and organizational data. We used the preprocessed versions of $sepsis\_cases\_2$ and $sepsis\_cases\_3$ logs made available in Pasquadibisceglie et al. (2021b). Some statistics of these logs are shown in Table 2.

We leveraged prefixes up to specified lengths (13 for $sepsis\_cases\_2$ and 22 for $sepsis\_cases\_3$) extracted from the original traces to train our predictive models. Each of these prefixes was assigned a specific label. To label each prefix, we adhered to the same strategy adopted by Teinemaa et al. (2019), where prefixes were classified as either *deviant* (label = 1) or *normal* (label = 0) based on some labelling criterion. Specifically, in the $sepsis\_cases\_2$, any trace (and all prefixes derived from it) was marked as deviant if the patient was eventually admitted to the intensive care unit, regular otherwise. For $sepsis\_cases\_3$, a deviant label was assigned to those traces corresponding to patients whose hospital discharge deviated from the most common discharge protocol, referred to as 'Release A'.

We employed the AE encoding from Section 2 to flatten each prefix trace.

6.2 Testbed

Experimental design We ran Algorithm 1 and tested it across various ensemble combination methods, with the number of Active Learning (AL) steps m varying from 0 to 8 –we did not consider higher values of m because they were empirically found not to improve accuracy performance appreciably (see Fig. 2 and associated comments), yet entailing heavier intervention from human experts. We set other parameters as follows: initial learning rate $lr = 0.001$, number of training epochs $e = 32$, per-step AL budget $b = 20$, number of base models $k = 5$, and validation set percentage $val\_perc = 10\%$.

In our simulated AL scenario, an expert is available for up to 8 days, with the capacity to label $b = 20$ traces daily. This results in a maximum of $b_T = b \times m = 160$ samples. The expert's role was emulated using an oracle-like procedure by unveiling the ground-truth labels of the selected b tuples.

We tested the algorithm with all possible ensembling modes, i.e., MAX, AVG, MEDIAN, SOUP and NONE, the latter of which actually consists in training only one base DPM of the form described in Section 5.2; hereinafter, these different configurations of the algorithm will be simply referred to as *ensemble_max*, *ensemble_avg*, *ensemble_median*, *ensemble_soup* and *single-model*, respectively.

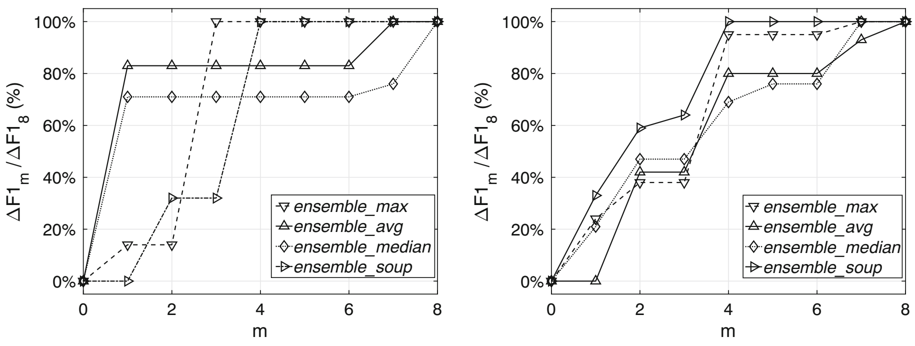


Fig. 2 Ratio between the F1's gain obtained in the offline setting by our approach, using fixed budget of $b = 20$, after $m \in [0, \ldots, 8]$ AL iterations and that obtained after 8 AL iterations on $\texttt{BPI}_{dM13}$ (left) and $\texttt{BPI}_{dM16}$ (right)

A train-test split for each dataset was executed by reserving a 20% sample D_{TEST} of instances to evaluate the models discovered by Algorithm 1 and its competitors. Specifically, a random selection method was used in the offline setting to split the $\texttt{BPI}_{dM13}$ and $\texttt{BPI}_{dM16}$ datasets. In contrast, we adopted a temporal split for `sepsis_cases_2` and `sepsis_cases_3` in the online setting, aligning with the experimental design of Teinemaa et al. (2019) and Pasquadibisceglie et al. (2021b).

The remaining 80% of training instances were then randomly divided into two equally-sized subsets. These were used to instantiate the sets D^L and D^U of Algorithm 1, with D^U serving as a set of instances with undisclosed (though in reality, hidden) class labels from which samples could be extracted during the AL procedure.

Evaluation metrics The accuracy of all DPMs was evaluated by resorting to three well-known metrics: *AUC*, *G-Mean*, and *F1* score. Since our tests involved imbalanced data (cf. Table 2), the *G-Mean* and *F1* scores are especially relevant as they are better suited in the case of uneven class distributions. In particular, in the following analysis, we will primarily rely on the *F1* as our metric of choice.

6.3 Quantitative results for offline deviance detection

In this section, we comprehensively analyze the results obtained in the offline experimental setting. In particular, Table 3 elucidates the performance on different metrics across two datasets and several AL iterations under various ensembling modes. A picture of the F1 gain achieved through our approach is shown in Fig. 2, which portrays its progression across different AL steps compared to the final iteration. Table 4 presents a comparative evaluation of our best-performing ensemble solutions, positioning them against a simplified (non-ensemble-based) approach and several cutting-edge models from recent literature.

A few noteworthy trends become evident when examining the results in Table 3. The first is that *ensemble_soup* typically stands out as a better option in the deviance prediction task, especially when considering the $F1$ and $G\text{-}Mean$ metrics, better suited for evaluating imbalanced logs like the ones at hand. This is particularly evident in the $\texttt{BPI}_{dM13}$ dataset. In the same log, despite *ensemble_max* yields a marginally higher AUC, the increase over *ensemble_soup* is merely +0.7%, underscoring the latter's overall robustness.

Inspecting the $\texttt{BPI}_{dM16}$ dataset, *ensemble_soup* still excels in $F1$ scores after all AL steps ($m > 0$) and keeps pace with (or slightly lags behind) other ensemble modes in terms of $G\text{-}Mean$ and AUC scores.

Interestingly, *ensemble_soup* is not the only standout. Excluding *ensemble_soup*, *ensemble_avg* merits special recognition for its consistent performance. Across various AL iterations and datasets, it consistently achieves top-tier results, establishing it as another reliable and effective strategy among other ensemble methods. However, the strength of *ensemble_soup* extends beyond performance metrics alone. As Wortsman et al. (2022) points out, *ensemble_soup* eliminates the need for maintaining and executing multiple concurrent models during the inference phase. This distinctive advantage reduces computational and memory demands, strengthening the case for *ensemble_soup* as a compelling strategy for this setting.

Regardless of the chosen ensembling mode, Table 3 showcases the effectiveness of our AL strategy in enhancing an ensemble DPM model over time. Indeed, using the budget b_T to add into D^L 160 properly selected traces from D^U over eight AL iterations significantly

Table 3 Results obtained, using fixed budget of $b = 20$, with different ensembling modes after varying numbers of AL iterations (i.e., different settings of hyperparameter m in Algorithm 1). The extreme setting $m = 0$ corresponds to using only the labelled data in the training set, with no actual AL iteration. Best results in bold

m	Model	BPI_{dM13}			BPI_{dM16}		
		AUC	$G\text{-}Mean$	$F1$	AUC	$G\text{-}Mean$	$F1$
0	*ensemble_max*	0.818	0.716	0.581	0.862	0.729	0.490
	ensemble_avg	0.803	0.689	0.548	0.877	**0.791**	**0.579**
	ensemble_median	0.807	0.687	0.544	**0.878**	**0.791**	**0.579**
	ensemble_soup	**0.822**	**0.740**	**0.614**	0.872	0.754	0.574
1	*ensemble_max*	0.813	0.723	0.586	**0.877**	0.759	0.526
	ensemble_avg	**0.823**	**0.752**	**0.619**	**0.877**	0.791	0.579
	ensemble_median	0.817	0.747	0.614	0.872	**0.799**	0.602
	ensemble_soup	0.822	0.740	0.614	0.853	0.793	**0.615**
2	*ensemble_max*	0.813	0.723	0.586	0.884	0.769	0.548
	ensemble_avg	0.823	0.752	0.619	0.888	0.795	0.621
	ensemble_median	0.817	0.747	0.614	0.883	0.805	0.629
	ensemble_soup	**0.831**	**0.759**	**0.631**	**0.891**	**0.797**	**0.646**
3	*ensemble_max*	**0.854**	0.745	0.617	0.884	0.769	0.548
	ensemble_avg	0.823	0.752	0.619	**0.888**	0.795	0.621
	ensemble_median	0.817	0.747	0.614	0.883	0.805	0.629
	ensemble_soup	0.831	**0.759**	**0.631**	0.878	**0.815**	**0.653**
4	*ensemble_max*	**0.854**	0.745	0.617	0.900	**0.814**	0.636
	ensemble_avg	0.823	0.752	0.619	0.899	0.810	0.660
	ensemble_median	0.817	0.747	0.614	**0.902**	**0.815**	0.653
	ensemble_soup	0.848	**0.785**	**0.667**	0.890	**0.814**	**0.697**
5	*ensemble_max*	**0.854**	0.745	0.617	**0.900**	**0.814**	0.636
	ensemble_avg	0.823	0.752	0.619	0.899	0.810	0.660
	ensemble_median	0.817	0.747	0.614	0.886	0.810	0.660
	ensemble_soup	0.848	**0.785**	**0.667**	0.890	**0.814**	**0.697**
6	*ensemble_max*	**0.854**	0.745	0.617	**0.900**	**0.814**	0.636
	ensemble_avg	0.823	0.752	0.619	0.899	0.810	0.660
	ensemble_median	0.817	0.747	0.614	0.886	0.810	0.660
	ensemble_soup	0.848	**0.785**	**0.667**	0.890	**0.814**	**0.697**
7	*ensemble_max*	**0.854**	0.745	0.617	0.887	0.830	0.643
	ensemble_avg	0.838	0.757	0.634	0.888	0.837	0.673
	ensemble_median	0.844	0.743	0.619	**0.899**	**0.849**	0.686
	ensemble_soup	0.848	**0.785**	**0.667**	0.890	0.814	**0.697**
8	*ensemble_max*	**0.854**	0.745	0.617	0.887	0.830	0.643
	ensemble_avg	0.838	0.757	0.634	0.887	0.832	0.680
	ensemble_median	0.847	0.763	0.643	**0.899**	**0.849**	0.686
	ensemble_soup	0.848	**0.785**	**0.667**	0.890	0.814	**0.697**

Table 4 Offline deviance mining: comparing our top-performing DPM ensembles, *ensemble_soup* and *ensemble_avg* with the *single-model* in two settings: (i) *No-AL*, using only the labelled data ($m = 0$), and (ii) *AL*, where a number $m \in \{4, 8\}$ of active learning iterations are performed after training the model over the labelled data only. As a term of comparison, the results of fully supervised (FS) state-of-the-art methods are reported for the ideal scenario where the deviance labels are disclosed for all the log traces (i.e., $D^U = \emptyset$ and $D = D^L$)

Setting	Model	BPI_{dM13}			BPI_{dM16}		
		AUC	$G\text{-}Mean$	$F1$	AUC	$G\text{-}Mean$	$F1$
No-AL	*ensemble_soup* ($m = 0$)	0.822	0.740	0.614	0.872	0.754	0.574
	ensemble_avg ($m = 0$)	0.803	0.689	0.548	0.877	0.791	0.579
	single-model ($m = 0$)	0.804	0.719	0.593	0.847	0.813	0.647
AL	*ensemble_soup* ($m = 4$)	0.848	0.785	0.667	0.890	0.814	0.697
	ensemble_avg ($m = 4$)	0.823	0.752	0.619	0.899	0.810	0.660
	single-model ($m = 4$)	0.831	0.728	0.606	0.861	0.802	0.660
	ensemble_soup ($m = 8$)	0.848	0.785	0.667	0.890	0.814	0.697
	ensemble_avg ($m = 8$)	0.838	0.757	0.634	0.887	0.832	0.680
	single-model ($m = 8$)	0.800	0.757	0.630	0.870	0.820	0.667
FS	$HO\text{-}DPM\text{-}mine$ (Cuzzocrea et al., 2016b)	0.841	0.741	0.633	0.878	0.778	0.643
	$MVDE\text{-}Stack$ (Folino et al., 2020)	0.878	0.801	0.675	0.874	0.815	0.608
	$MVDE\text{-}Max$ (Folino et al., 2020)	0.864	0.654	0.566	0.907	0.668	0.545

improves the performance of the DPM ensemble models across all metrics and datasets. Notably, when $m = 8$, *ensemble_median* shows an improvement in terms of AUC, G-$Mean$, and $F1$ of approximately 5%, 11%, and 18% on the $\texttt{BPI}_{dM13}$ dataset, and 2%, 7%, and 18% on the $\texttt{BPI}_{dM16}$ dataset if compared to the non-AL-enhanced setting $m = 0$. Similar improvements are observed on the $\texttt{BPI}_{dM16}$ dataset for the *ensemble_avg*, *ensemble_soup*, and *ensemble_max* strategies —these latter ensemble strategies seem to benefit less from the AL approach on $\texttt{BPI}_{dM13}$.

Further insights on the effectiveness of the AL procedure can be drawn from Fig. 2, which illustrates the ratio $\Delta F1_m\%/\Delta F1_8\%$ for $m = 0, \ldots, 8$. This ratio represents the performance difference (in terms of F1) between the DPM ensemble obtained after m AL iterations and that at the last iteration. This figure helps us see that after just 3-4 AL iterations (i.e., using half the expert budget $b_T = 160$) all DPM ensembles perform nearly equivalently to their respective fully-grown versions discovered after $m = 8$ AL steps. For example, on dataset $\texttt{BPI}_{dM13}$ (Fig. 2, left), if making the expert label only 60 or 80 traces (i.e. 13% or 18% of the traces in D^U, respectively), *ensemble_max* and *ensemble_soup* would reach, in 4 AL iterations, 100% of the F1 score obtained once exploited the full budget b_T. The *ensemble_avg* and *ensemble_median* modes reach 81% and 72% of the $F1$ value when $m = 8$, respectively. Comparable results are seen with dataset $\texttt{BPI}_{dM16}$ (Fig. 2, right).

Figure 3 highlights the relationship between the budget b used in the AL cycles and the model's predictive performance. The model's predictive capabilities improve as b grows, providing more instances for expert evaluation. This trend is reasonably expected, given that strategic tuple selection enables training more accurate models. Anyway, the results achieved with the (relatively low) default budget value of $b = 20$ are satisfactory enough and not so far from those obtained when using twice that budget. This means that $b = 20$ can be regarded as an option that ensures an acceptable trade-off between the accuracy and the practical usability and sustainability of the approach. In fact, in real-world scenarios, there are limitations to how many instances an expert can feasibly label. For example, while labelling 20 instances daily

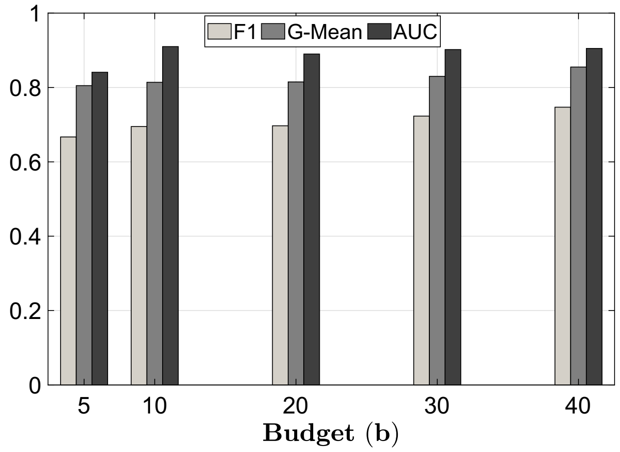


Fig. 3 Trend of the three performance metrics (AUC, G-Mean, and F1) obtained by model *ensemble_soup* when $m = 8$ across different settings of the budget hyperparameter b (namely, $b = 5, 10, 20, 30, 40$) on dataset $\texttt{BPI}_{dM16}$

might be manageable, expecting an expert to label twice that amount becomes unrealistic. This practical consideration is precisely why we have chosen a budget of $b = 20$.

Finally, as we look at Table 4, we find a comparison of results from our best-performing ensemble models *ensemble_soup* and *ensemble_avg* (as discerned from Table 3) against those obtained by the following DPM discovery methods: (1) a non-ensemble-based approach, denoted as *single-model*, that consists in training (just one instance of) the base DPM architecture of Section 5.2 against the labelled set D^L at the $m = 0, 4, 8$ iterations of the AL procedure encoded by Algorithm 1; and (2) three state-of-the-art multi-view ensembling approaches, namely method *HO-DPM-mine* (Cuzzocrea et al., 2016b) and two variants of method *MVDE* (Folino et al., 2020), evaluated in an ideal fully-supervised (FS) scenario where the class labels of all the instances in $D^L \cup D^U$ are exploited to train the DPM models. In contrast, *ensemble_soup* and *ensemble_avg* only use the labels associated with the instances stored in D^L and the few iteratively selected from D^U during the AL procedure.

Interestingly, from Table 4, *ensemble_soup* shows remarkable consistency across all AL stages and data scenarios. For instance, at the beginning step $m = 0$, it surpasses *single-model* (and *ensemble_avg*) in all metrics on the $\texttt{BPI}_{dM13}$ dataset. Even though the simpler *single-model* scores slightly higher on the $\texttt{BPI}_{dM16}$ dataset at the same stage, *ensemble_soup* still maintains solid performances. As m increases to 4 and 8, we observe performance enhancements for both *ensemble_soup* and *ensemble_avg* across all metrics. This advancement is notably more pronounced than that of *single-model*, highlighting the convenience of continuing the AL procedure in our ensemble-based DPMs. By the final AL iteration at $m = 8$, *ensemble_avg* shows minor improvements over *ensemble_soup* in the *G-Mean* score on $\texttt{BPI}_{dM16}$.

Overall, *ensemble_soup* and *ensemble_avg* typically deliver performance superior to *single-model* across diverse data scenarios, while maintaining the same learning cost. In addition, even amidst occasional fluctuations where *ensemble_avg* edges out, the overall performances of *ensemble_soup* remain quite competitive by ensuring no additional computational and memory overheads associated with snapshot-based ensemble models. These performances make *ensemble_soup* a particularly appealing choice for the deviance prediction task within an AL setting, especially when high-stakes efficiency requirements call to be met.

The comparison with fully-supervised methods in Table 4 highlights the surprising competitiveness of our ensemble methods. Indeed, at the end of the AL process, *ensemble_soup* and *ensemble_avg* manage to hold their ground and occasionally outperform these advanced models. This advantage is notable on both $\texttt{BPI}_{dM13}$ and $\texttt{BPI}_{dM16}$ datasets when evaluated with the important metrics *G-Mean* and $F1$ —the comparison in terms of AUC is, instead, slightly less favourable.

Delving into specifics, *ensemble_soup* and *ensemble_avg* consistently outperform *HO-DPM-mine* and *MVDE-Max*, but fall a bit short of *MVDE-Stack* on the $\texttt{BPI}_{dM13}$ dataset. However, it is worth highlighting that *MVDE-Stack*, besides being multi-view, employs a trainable (thus, more complex and costlier) stacking-based function to merge predictions from all base models. In contrast, *ensemble_soup* merely amalgamates the weights of different steps of the base DPM model to yield a single one, while *ensemble_avg* straightforwardly averages the predictions of all DPM snapshots. These markedly more efficient approaches empower them to balance performance and computational efficiency optimally. This balance becomes especially significant in scenarios where DPM ensembles require frequent updates.

6.4 Quantitative results for online deviance prediction

Table 5 seeks to examine the effectiveness of the *ensemble_soup* and *ensemble_avg* models, previously assessed in Table 4, when they cope with a predictive monitoring task, that is significantly more demanding than offline deviance prediction. The focus here is to evaluate their ability to forecast whether an ongoing trace is deviant using two prefix datasets extracted from the original *sepsis_cases*_2 and *sepsis_cases*_3 logs. Similarly to the offline setting, the comparison involves the *single-model* within the AL context and other renowned predictive models from existing literature in the FS context, including the ensemble-based *XGBoost* and *Random Forest* models from Teinemaa et al. (2019) and the fuzzy model FOX (Pasquadibisceglie et al., 2021b).

Examining Table 5, it is clear that *ensemble_soup* and *ensemble_avg* reap substantial benefits from the AL process, generally exceeding that of the *single-model.* These models display an upward performance trajectory as the AL iterations increase in both datasets. This trend confirms the observed in the offline setting that AL strategy progressively enhances our models' predictive ability.

Looking deeper at case m=0, one sees that both *ensemble_avg* and *ensemble_soup* beat *single-model* in all metrics on *sepsis_cases*_2, but *ensemble_soup* gets slightly worse AUC and $F1$ scores on *sepsis_cases*_3 than *single-model.*

Comparing the ensemble models, *ensemble_avg* outperforms *ensemble_soup* in AUC on *sepsis_cases*_2, whereas *ensemble_soup* excels in $F1$ on the same dataset. On *sepsis_cases*_3, *ensemble_avg* leads over *ensemble_soup* in both these metrics. This landscape changes at $m = 4$. At this step, *ensemble_avg* surpasses both the *single-model* and *ensemble_soup* in all performance measures on *sepsis_cases*_3 and in terms of *G-Mean* and $F1$ on *sepsis_cases*_2. Conversely, *ensemble_soup* surpasses the *single-model* in all metrics for *sepsis_cases*_2 but lags in *G-Mean* and $F1$ on *sepsis_cases*_3. At the last AL step $m = 8$, both models' performance plateau. Notably, *ensemble_avg* retains its advantage over *ensemble_soup* and *single-model*, except for AUC on *sepsis_cases*_2 where *ensemble_soup* leads. These results suggest that despite minor performance fluctuations, both *ensemble_avg* and *ensemble_soup* consistently leverage the AL process to enhance their predictive performance over time, with *ensemble_avg* stably surpassing the *single-model* across all metrics and datasets, and *ensemble_soup* outperforming *single-model* in *sepsis_cases*_2.

Considering the less favourable FS setting in Table 5, while *ensemble_avg* and *ensemble_soup* do not achieve the highest metric values, they perform satisfactorily compared to the competitors. For example, at the intermediate AL step $m = 4$, *ensemble_avg* already surpasses both *XGBoost* and *Random Forest* in terms of $F1$ on *sepsis_cases*_2 and significantly outperforms (+249%) them on *sepsis_cases*_3 in the same metric. In comparison, *ensemble_soup* holds a similar considerable gain on $F1$ on *sepsis_cases*_3 but yields some ground on the same metric on *sepsis_cases*_2. Both ensemble models perform worse than their competitors in the AUC measure.

Though *XGBoost* and *Random Forest* yield comparable, or slightly better, $F1$ scores in the FS setting on *sepsis_cases*_2 and a more noticeable increase in AUC across both datasets, achieving such performance typically requires using a substantial high number (sometimes hundreds) of base learners. This highlights the efficiency of the learning strategy adopted in *ensemble_soup*, which achieves good performances by combining only five base models while reducing the computational and memory costs associated with traditional ensemble-based learning approaches. This effectiveness-efficiency trade-off is relevant in an online predictive context.

Table 5 Online deviance mining: comparing our top-performing DPM ensembles, $ensemble\_soup$ and $ensemble\_avg$ with the *single-model* in two settings: (i) *No-AL*, using only the labeled data ($m = 0$), and (ii) *AL*, where a number $m \in \{4, 8\}$ of Active Learning iterations are performed after training the model over the labelled data only. As a term of comparison, the results of fully supervised (FS) state-of-the-art methods are reported for the ideal scenario where the deviance labels are disclosed for all the log traces (i.e., $D^U = \emptyset$ and $D = D^L$)

Setting	Model	sepsis_cases_2			sepsis_cases_3		
		AUC	$G\text{-}Mean$	$F1$	AUC	$G\text{-}Mean$	$F1$
No-AL	*ensemble_soup* ($m = 0$)	0.703	0.585	0.318	0.606	0.458	0.870
	ensemble_avg ($m = 0$)	0.741	0.479	0.311	0.614	0.433	0.893
	single-model ($m = 0$)	0.560	0.335	0.164	0.673	0.461	0.892
AL	*ensemble_soup* ($m = 4$)	0.726	0.547	0.336	0.604	0.406	0.898
	ensemble_avg ($m = 4$)	0.707	0.600	0.415	0.660	0.467	0.904
	single-model ($m = 4$)	0.688	0.503	0.331	0.578	0.441	0.900
	ensemble_soup ($m = 8$)	0.726	0.547	0.336	0.604	0.406	0.898
	ensemble_avg ($m = 8$)	0.707	0.600	0.415	0.688	0.476	0.908
	single-model ($m = 8$)	0.688	0.503	0.331	0.578	0.441	0.900
FS	$XGBoost$ (Teinemaa et al., 2019)	0.800	–	0.400	0.710	–	0.210
	$Random\ Forest$ (Teinemaa et al., 2019)	0.760	–	0.410	0.700	–	0.260
	FOX (Pasquadibisceglie et al., 2021b)	0.730	–	–	0.680	–	–

Comparing our ensemble models at the last AL iteration with FOX yields interesting observations. The performance of $ensemble\_avg$ and $ensemble\_soup$ is indeed comparable with that of FOX in terms of AUC, except for $sepsis\_cases\_3$, where $ensemble\_soup$ scores lower. However, despite the renowned fuzzy models' efficiency in terms of memory and computation, FOX comes with a considerable number of rules (Pasquadibisceglie et al., 2021b) (729 and 81 for $sepsis\_cases\_3$ and $sepsis\_cases\_2$, respectively), causing a computational overhead possibly offsetting its efficiency benefits when compared to our ensemble models.

In predictive monitoring, a key property of a predictive model consists in its ability to forecast outcomes at early stages accurately enough. To evaluate this property for the DPMs discovered by our approach, we computed the average AUC score achieved by the $ensemble\_soup$ variant of the approach (at its final AL iteration, i.e. $m = 8$) against different sub-sets of the test traces of datasets $sepsis\_cases\_2$ and $sepsis\_cases\_3$, grouped by prefix length. Figure 4 illustrates the progression of these AUC scores over varying prefix lengths for both datasets.

For both datasets, the AUC trend is in line with what expected: the score is acceptable (well above 0.5) for short prefixes and tends to increase as more information becomes available, i.e., as the prefix length gets longer than 5 or 10, respectively in the two cases. A performance peak is reached at the 9th and the 13th execution steps for $sepsis\_cases\_2$ and $sepsis\_cases\_3$, respectively. Then, beyond these steps, the AUC tends to slightly decline –a trend that looks more pronounced on $sepsis\_cases\_3$.

At first glance, this may seem counter-intuitive, as longer prefixes are more informative. Yet, similar AUC trends were seen in previous studies like Teinemaa et al. (2019). Notably, the observed AUC dip for longer prefixes does not necessarily indicate a model's flaw. In our opinion, this decline can be attributed to a lack of large and homogeneous enough subsets of prefixes with those higher lengths, since, as posited in Teinemaa et al. (2019), most of the fully-grown traces are far shorter and predicting the outcome of their associated prefixes is an easy task even if these prefixes consist of few events (but not so far from the last event of the process instance).

6.5 Qualitative results: deviance prediction explanations

To help the user understand and critically analyze the factors that could have led a discovered DPM (or DPM ensemble) to predict a test instance as deviant/normal, in the prototypal

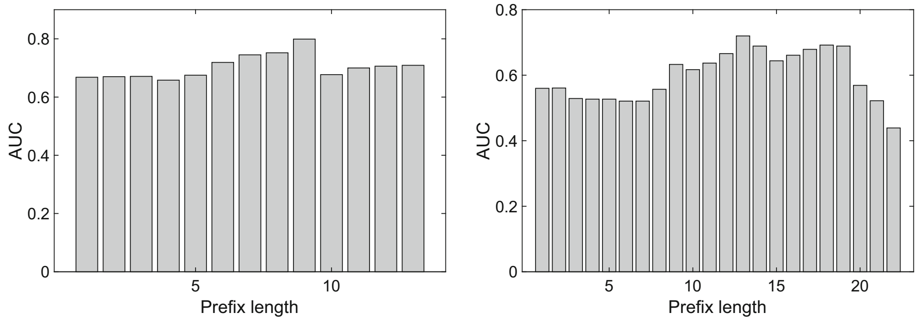


Fig. 4 Average AUC scores, for varying prefix lengths, of the $ensemble\_soup$ models found (with $m = 8$) for datasets $sepsis\_cases\_2$ (left) and $sepsis\_cases\_3$ (right)

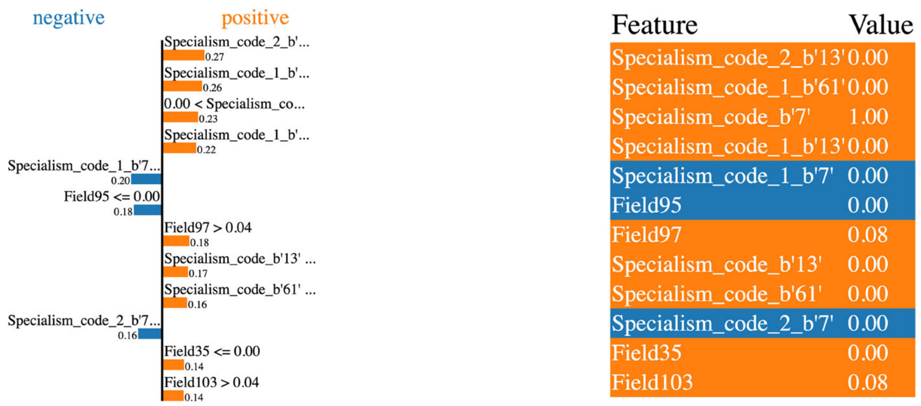
implementation of our approach, a public Python implementation of the post-hoc explanation method *LIME* (Ribeiro et al., 2016) was integrated. Though different frameworks (e.g., *SHAP* and *Grad-CAM*) have been proposed in the literature to make interpretable black-box models, LIME is adopted in our solution to limit the computational costs required to yield the explanations.

Figure 5 shows an example of an explanation obtained with LIME about why traces are deemed deviant by our ensemble model when applied to the dataset BPI_{dM16}. Basically, it seems that the prediction of deviance/normality is mainly concerned with specific values of some event attributes in the family of "specialism code" (i.e., `Specialism code`, `Specialism code 1` and `Specialism code 2`), and with the presence/absence of specific patterns (either IA or DP) in the data —all patterns (very long sets of attributes, potentially) have been mapped, for the sake of readability, with an enumerated field (e.g., `Field95`, `Field97` etc.) in the figure.

In particular, from Fig. 5 it emerges that the presence of value '7' for the attribute `Specialism code` as well as the absence of value '13' (resp., '61' and '13') for the attribute `Specialism code 2` (resp., `Specialism code 1`) are all positively correlated with the prediction of deviance for the trace at hand. By contrast, the absence of the value '7' for the attribute `Specialism code 1` and that of the pattern `Field95` negatively correlates with the prediction of deviance for the same trace. Further explanation artefacts and analyses are not reported here for lack of space.

Figure 6 is meant to summarize some insights derived from DPM ensemble's predictions on dataset $\texttt{BPI}_{dM16}$. This chart emphasizes the features exhibiting the strongest positive and negative correlations to the deviance class. For the sake of clarity, we here focus on the top 10 features per correlation direction.

As seen in Fig. 5, LIME's strength mainly lies in its ability to offer local explanations. It does so by attributing a relevance score to features that primarily impact the prediction of a particular process trace. To roughly quantify the global impact of the data features, we averaged the scores returned by LIME across all traces in the test set. This allowed us to obtain a (sort-of) global relevance score for each feature, like those shown in Fig. 6. Based on these aggregated results, the features playing as main prediction drivers include various "specialism codes", patients' ages, and a series of attributes encapsulated within distinct enumerated fields.



Feature	Value
Specialism_code_2_b'13'	0.00
Specialism_code_1_b'61'	0.00
Specialism_code_b'7'	1.00
Specialism_code_1_b'13'	0.00
Specialism_code_1_b'7'	0.00
Field95	0.00
Field97	0.08
Specialism_code_b'13'	0.00
Specialism_code_b'61'	0.00
Specialism_code_2_b'7'	0.00
Field35	0.00
Field103	0.08

Fig. 5 An example of LIME local explanation of a prediction made (with a DPM ensemble) for a trace of dataset BPI_{dM16}

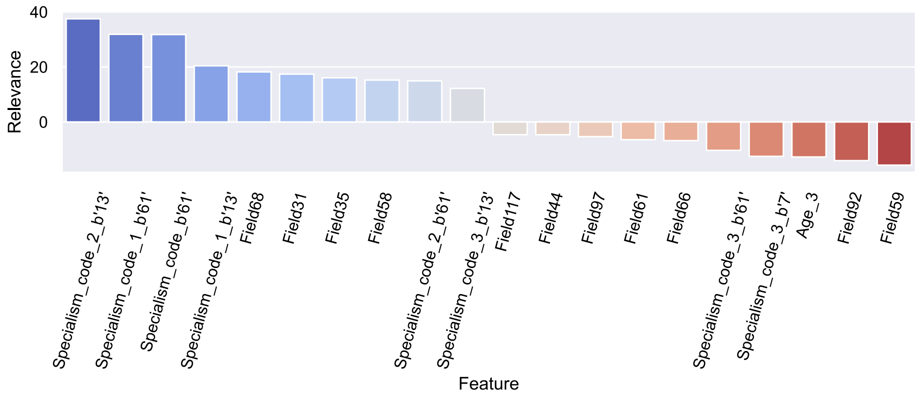


Fig. 6 Data features with the strongest (positive or negative) global influence on the deviance class, according to the DPM ensemble mined from dataset $\texttt{BPI}_{dM16}$. Positive and negative influence scores are shown in blue and red, respectively

7 Conclusion and future work

An AL-based framework for discovering a deep DPM was proposed. The approach employs a temporal ensembling method for training multiple base DPMs (sharing a DNN architecture featuring dropout and residual-like components) and fusing them according to different alternative strategies concerning either model outputs or model parameters. The latter option, in particular, is exploited by the *soup* variant of the framework, which is meant to save memory and compute costs. Test results obtained on real-life log data for both completed and ongoing process instances demonstrate the method's effectiveness, even compared to the results obtained by state-of-the-art supervised algorithms in the ideal case where all the trace labels are known.

Some limits of our work concern the lack of: (i) a deep analysis (e.g., based on a user study) of the practical utility of the features' impact scores found in the tests, and (ii) tests performed with alternative trace encoding and uncertainty estimation methods. As to future work, besides bridging these two gaps, we plan to: (1) study how the features' impact changes across different prefix lengths; (2) evaluate the framework on streaming logs; and (3) extend it with data/model distillation capabilities.

Acknowledgements This work was partially supported by projects SERICS (PE00000014) and FAIR (PE00000013) funded by the EU under the NGEU program (and the associated NRRP MUR Italian program), and by project PINPOINT funded by Italian ministry MUR (grant no. B27G22000160001) under program PRIN.

Author Contributions These authors contributed equally to all the different parts of this work.

Funding Open access funding provided by Consiglio Nazionale Delle Ricerche (CNR) within the CRUI-CARE Agreement.

Data Availability The data are publicly accessible (see associated references).

Declarations

Ethical Approval Not applicable.

Competing interests The authors declare no competing interests.

References

Adadi, A. (2021). A survey on data-efficient algorithms in big data era. *Journal of Big Data, 8*, 24.

Atzmueller, M. (2015). Subgroup discovery - advanced review. *Wiley Intl Rev Data Min and Knowl Disc, 5*(1), 35–49.

Bose, R. P. J. C., van der Aalst W. M. P. (2013) Discovering signature patterns from event logs. In: IEEE Symp. on Comput. Intell. and Data Mining (CIDM'13), pp 111–118

Cuzzocrea, A., Folino, F., Guarascio, M., et al. (2015) A multi-view learning approach to the discovery of deviant process instances. In: OTM Confederated Intl. Conf.s" On the Move to Meaningful Internet Systems", Springer, pp 146–165

Cuzzocrea, A., Folino, F., Guarascio M, et al. (2016a) A multi-view multi-dimensional ensemble learning approach to mining business process deviances. In: 2016 Intl. Joint Conf. on Neural Networks (IJCNN), pp 3809–3816

Cuzzocrea A, Folino F, Guarascio M, et al (2016b) A robust and versatile multi-view learning framework for the detection of deviant business process instances. *International Journal of Cooperative Information Systems25*(04):1740,003

Di Francescomarino, C., & Ghidini, C. (2022) *Predictive process monitoring. Process Mining Handbook* pp 320–346

Fani Sani, M., van der Aalst, W., Bolt, A., et al. (2017) Subgroup discovery in process mining. In: Business Information Systems: 20th Intl. Conf., BIS 2017, Poznan, Poland, June 28–30, 2017, Proceedings, Springer, p 237

Fazzinga, B., Flesca, S., Furfaro, F., et al. (2018). Online and offline classification of traces of event logs on the basis of security risks. *J Intell Inf Syst, 50*(1), 195–230.

Folino, F., & Pontieri, L. (2019) Business process deviance mining. In: *Encyclopedia of Big Data Technologies*. Springer

Folino, F., Folino, G., Guarascio, M., et al. (2020) A multi-view ensemble of deep models for the detection of deviant process instances. In: *ECML-PKDD Worksh*, pp 249–262

Folino, F., Folino, G., Guarascio, M., et al. (2022) Combining active learning and fast DNN ensembles for process deviance discovery. In: Proc. of 26th Intl Symp. on Foundations of Intelligent Systems (ISMIS'22), pp 346–356

Folino, F., Guarascio, M., Pontieri, L. (2017) A descriptive clustering approach to the analysis of quantitative business-process deviances. In: Proceedings of the 32nd ACM SIGAPP Symposium on Applied Computing (SAC'17), ACM, pp 765–770

Folino, F., Guarascio, M., & Pontieri, L. (2013). Context-aware predictions on business processes: An ensemble-based solution. *New Frontiers in Mining Complex Patterns* (pp. 215–229). Berlin Heidelberg, Berlin, Heidelberg: Springer.

Huang, G., Li, Y., Pleiss, G., et al. (2017) Snapshot ensembles: Train 1, get M for free. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings

Lo, D., Cheng, H., Han, J., et al. (2009) Classification of software behaviors for failure detection: A discriminative pattern mining approach. In: Proc. of 15th Int. Conf. on Knowledge Discovery and Data Mining (KDD'09), pp 557–566

Ly, L. T., Maggi, F. M., Montali, M., et al. (2015). Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems, 54*, 209–234.

Mannhardt, F. (2016). *Sepsis cases - event log*. https://doi.org/10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460

Neu, D. A., Lahann, J., & Fettke, P. (2022). A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artif Intell Rev, 55*(2), 801–827.

Neyshabur, B., Sedghi, H., Zhang, C. (2020) What is being transferred in transfer learning? In: Advances in Neural Information Processing Systems, pp 512–523

Nguyen, H., Dumas, M., Rosa, M. L., et al. (2014) Mining business process deviance: A quest for accuracy. In: In Proc. of OTM 2014 Conferences, pp 436–445

Pasquadibisceglie, V., Appice, A., Castellano, G., et al. (2021). A multi-view deep learning approach for predictive business process monitoring. *IEEE Transactions on Services Computing, 15*(4), 2382–2395.

Pasquadibisceglie V, Castellano G, Appice A, et al. (2021b) Fox: a neuro-fuzzy model for process outcome prediction and explanation. In: 2021 3rd International Conference on Process Mining (ICPM), pp 112–119

Ren, P., Xiao, Y., Chang, X., et al. (2021). A survey of deep active learning. *ACM Computing Surveys (CSUR), 54*(9), 1–40.

Ribeiro, M., Singh, S., Guestrin, C. (2016) “why should I trust you?”: Explaining the predictions of any classifier. In: Proc. of 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp 97–101

Rinderle-Ma, S., & Winter, K. (2022). Predictive compliance monitoring in process-aware information systems: State of the art, functionalities, research directions. *Inf Syst, 115*(102), 210.

Suriadi S, Wynn MT, Ouyang C, et al. (2013) Understanding process behaviours in a large insurance company in australia: A case study. In: Proc of 25th Int. Conf. on Advanced Information Systems Engineering (CAiSE’13), pp 449–464

Swinnen, J., Depaire, B., Jans, M. J., et al. (2012) A process deviation analysis–a case study. In: Proc. of Intl. Conf. on Business Process Management, pp 87–98

Teinemaa, I., Dumas, M., La Rosa, M., et al. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD), 13*(2), 1–57.

van Dongen, B. (2011). *Real-life event logs - hospital log.* https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

Wortsman, M., Ilharco, G., Gadre, S. Y., et al. (2022) Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Proceedings of Machine Learning Research*, *162*