Bus Inf Syst Eng 64(6):729–749 (2022)
https://doi.org/10.1007/s12599-022-00749-9

RESEARCH PAPER

Semi-Supervised Discovery of DNN-Based Outcome Predictors from Scarcely-Labeled Process Logs

Francesco Folino · Gianluigi Folino · Massimo Guarascio · Luigi Pontieri

Received: 23 October 2020 / Accepted: 3 February 2022 / Published online: 1 April 2022

Abstract Predicting the final outcome of an ongoing process instance is a key problem in many real-life contexts. This problem has been addressed mainly by discovering a prediction model by using traditional machine learning methods and, more recently, deep learning methods, exploiting the supervision coming from outcome-class labels associated with historical log traces. However, a supervised learning strategy is unsuitable for important application scenarios where the outcome labels are known only for a small fraction of log traces. In order to address these challenging scenarios, a semi-supervised learning approach is proposed here, which leverages a multi-target DNN model supporting both outcome prediction and the additional auxiliary task of next-activity prediction. The latter task helps the DNN model avoid spurious trace embeddings and overfitting behaviors. In extensive experimentation, this approach is shown to outperform both fully-supervised and semi-supervised discovery methods using similar DNN architectures across different real-life datasets and label-scarce settings.

Keywords Process mining · Outcome prediction · Deep learning · Semi-supervised learning

Accepted after two revisions by Stefan Lessmann

F. Folino · G. Folino · M. Guarascio · L. Pontieri (✉)
ICAR-CNR, Via Pietro Bucci 8-9C, 87036 Rende, CS, Italy
e-mail: luigi.pontieri@icar.cnr.it

F. Folino
e-mail: francesco.folino@icar.cnr.it

G. Folino
e-mail: gianluigi.folino@icar.cnr.it

M. Guarascio
e-mail: massimo.guarascio@icar.cnr.it

1 Introduction

Temporally annotated event logs are generated and maintained in an increasing number of companies and organizations, as a precious kind of data concerning the execution of their business processes. In general, by analyzing these log data with Process Mining techniques (Van Der Aalst 2011), valuable information and models can be extracted that help better comprehend, monitor and handle the processes that generated them. Predictive (process) monitoring (Maggi et al. 2014) is a sub-field of Process Mining that aims at providing process monitoring frameworks with the ability of making forecasts for any ongoing process instance, say p, based on its associated trace, i.e., the sequence of events stored for p up to the moment of prediction. In particular, Outcome Prediction (Teinemaa et al. 2018, 2019; Metzger et al. 2019) amounts to predicting the outcome class of a process instance p, based on the current trace of p. Most of the existing approaches to outcome prediction rely on discovering a prediction model (or predictor, for short) from historical log traces by using Machine Learning (ML) methods. In particular, many state-of-the-art solutions (Kratsch et al. 2020; Mehdiyev et al. 2018; Tax et al. 2017; Evermann et al. 2017; Navarin et al. 2017; Hinkka et al. 2018; Teinemaa et al. 2018) in this context resort to deep neural networks (DNNs), which allow for achieving good accuracy results (Kratsch et al. 2020; Teinemaa et al. 2019) while requiring minor data engineering efforts.

1.1 Problem: Discover an Outcome Predictor with Few Labeled Traces

All the existing learning-based approaches to outcome prediction (apart from Folino et al. (2019), which is

discussed later on) assume that the outcome of every completed process instance p is known with certainty (and stored in the same information system as p). Unfortunately, the above-mentioned assumption does not hold in relevant real-life application scenarios where the kind of outcome to be predicted, for a process instance p, is related to security, compliance or quality properties that cannot be assessed in a fully automated and certain way, based on the data that are usually gathered during the execution of p. Some important scenarios of this kind are sketched below:

- *Outcomes based on customers' feedback*: Customers' feedback on the quality of a product/service, usually acquired through surveys, are taken into account to a great extent by modern enterprises and organizations. Based on this feedback, one can define customer-satisfaction-oriented outcome classes for the instances of various operating processes (ranging, e.g., from the production, testing and delivery of a product to service provision and to the handling of customers' complaints/requests), in order to eventually analyze, predict and improve the behavior of these process instances. However, this feedback information often suffers from high non-response rates and reliability concerns (Lin and Jones 1997), and needs careful assessment by experts. Moreover, technological obstacles may arise in merging new traces with their respective (customer-driven) outcome classes, when these two pieces of information are kept in different information silos. Consequently, in this scenario, small numbers of traces with associated outcome labels are available to train a model for predicting such a customer-related outcome accurately.
- *Auditing-dependent outcomes*: Several discovery approaches in the field (e.g., those in Maggi et al. (2014) and Teinemaa et al. (2019), respectively) were either devised for or applied to settings where the outcome variable to be predicted concerns the satisfaction of business goals/constraints, under the assumption that the values of this variable can be evaluated for every log trace. However, there are many real-life scenarios where such goals/constraints are not expressed in a precise way and/or are difficult to be encoded fully and correctly into a machine-readable form. This is common when it comes to evaluate the compliance to complex norms stated in jargon-rich text (e.g., regulatory laws) (Hashmi et al. 2018). Deciding the outcome class of a process instance in such a case entails labor-intensive (post-execution) auditing activities, typically performed by specialized consultants on small samples of log data (Chan and Vasarhelyi 2018). On the other hand, when the outcome class is meant to indicate the presence of security attacks (e.g., fraud, information theft/leakage) (Fazzinga et al. 2020), it is unrealistic to assume that there are rule-sets/models allowing for assigning the class label exactly to every log trace. In this case, one can use unsupervised anomaly detection tools (Nolle et al. 2018) to pre-classify the process instances automatically as either conforming/normal or deviant/insecure, but the resulting class assignments must undergo careful manual revisions, which entail costs (in terms of time and of resources required) that are too substantial to sustain on large amounts of log traces. Thus, in all these auditing-dependent contexts, a small number of outcome-labeled traces are usually available for training an outcome-prediction model.

The problem addressed: In this work, we want to address the problem of discovering an accurate and robust DNN-based outcome-prediction model in "label-scarce" scenarios like those described above, i.e., scenarios where the ground-truth outcome class is available only for a small fraction of the log traces that can be used to train the model. This problem is definitely important in the above-mentioned kinds of application scenarios, where the discovery of good outcome predictors would allow for implementing proactive run-time support mechanisms in order to improve the quality of a process instance (in terms of customer-satisfaction, compliance, or security), and for activating policies in order to mitigate the impact of undesired/deviating/insecure outcomes that can no longer be prevented.

1.2 Limitations of Existing Solutions

The discovery of outcome predictors has been faced so far as a supervised (machine) learning task, under the above-mentioned assumption that every log trace has an associated outcome label. In particular, almost all the existing approaches to learning a DNN-based predictor (including, e.g., Kratsch et al. (2020); Teinemaa et al. (2019); Hinkka et al. (2018); Metzger et al. (2019)) rely on training it only over the outcome-labeled traces (after splitting them into training and validation sets), using standard mini-batch gradient descend methods (Goodfellow et al. 2016). However, this strategy does not fit application scenarios like those discussed above, where the few labeled traces available hardly suffice for training a DNN adequately, considering the expressiveness of these models and the huge size of the search space (i.e., all possible configurations of the trainable/free parameters in the DNN). In fact, DL methods are known to be far more data-hungry (Liu et al. 2021) than classic ML ones, since they are meant to extract abstract features automatically from raw data (without the guidance of heavy manual feature-engineering

steps), and neural networks tend to have limited generalization ability (Xu et al. 2020). Thus, when trained on a small amount of labeled data, a DNN predictor is very likely to incur overfitting and to rely on contingent properties of the data instances in making predictions for them (Liu et al. 2021).

To the best of our knowledge, the only semi-supervised-learning (Van Engelen and Hoos 2020; Ouali et al. 2020) approach to outcome prediction has been proposed in Folino et al. (2019), where the problem of discovering an outcome predictor in a label-scarce scenario was originally stated. This approach consists in using a pure pre-training procedure, where a DNN model is (i) first trained over all the log traces (having an outcome label or not) to predict the next activity of a trace, and then (ii) fine-tuned over the outcome-labeled traces, in order to adapt the model to the task of interest (i.e., outcome prediction), named hereinafter the *target task*. In line with the emerging paradigm of Self-Supervised Learning (Liu et al. 2021), the auxiliary task of activity prediction is a "pretext" supervised task defined over all the trace prefixes (for each of which, the ground-truth next-activity label is known), which can be solved via standard gradient-descent optimization. Essentially, this auxiliary task aims at training the ("encoder" sub-net of) DNN to extract general embeddings (i.e., dense representations) for the traces, so reducing the risk of discovering an overfitting predictor. This simple pre-training strategy was empirically shown in Folino et al. (2019) to be more effective in a specific real-life label-scarce scenario, in comparison with the traditional fully-supervised learning approach.

However, the method in Folino et al. (2019) may well fail to find an optimal balance between the auxiliary and target tasks, which govern the pre-training and fine-tuning phases, respectively, in an isolated way. In particular, two risks threaten it: (a) the pre-training phase leads the DNN to a region of the parameter space that overfits the auxiliary task, and prevents the DNN from adapting well to the target task (a.k.a. representation degeneration), and (b) the representation-learning skills learned in the pre-training phase are completely lost in the fine-tuning one (a.k.a. catastrophic forgetting) (these risks are discussed in Sect. 5.2 in some more detail). In fact, the limited empirical study conducted in Folino et al. (2019) (on just a single process log), does not allow for assessing how robust to these risks this pre-training solution is in wider range of application contexts.

Let us remark that the peculiarities of process logs prevent reusing solutions like transferring knowledge learned in another label-rich domain or augmenting labeled data artificially, which were successfully employed in certain deep learning (DL) settings, to cope with the lack of labeled data (Ouali et al. 2020; Liu et al. 2021).[1]

1.3 Research Goals, Contribution and Organization

This research work stemmed from our desire of facing the outcome-predictor discovery problem stated before (in Sect. 1.1) by exploiting a novel semi-supervised strategy, different from the pure pre-training one of Folino et al. (2019) and, hopefully, more robust to the above-mentioned risks to which the latter is exposed. To this end, we specifically pursued two main research goals:

- RG': Devise a semi-supervised discovery method enabling a more synergistic integration between the target outcome-prediction task and the auxiliary next-activity prediction, compared to current semi-supervised solutions.
- RG'': Experimentally compare this proposed method with the other existing semi-supervised ones, assessing the ability of each method to improve a traditional supervised approach in situations where a limited fraction of training traces have an outcome label.

We approached RG' by defining a DNN-discovery algorithm, named $\texttt{SSOP}-\texttt{MTL}$, that trains a multi-target DNN in accomplishing the auxiliary and target tasks jointly, using a shared encoding sub-net to extract suitable trace embeddings for both tasks. This algorithm extends standard multi-objective mini-batch training schemes (Goodfellow et al. 2016) by including specific mechanisms to vary the relative weight of the two tasks during the training process, so that the learning bias is made pass from one task to the other in a gradual way. This facilitates a harmonized interplay between the two, somewhat diverging, objectives of ensuring accurate outcome predictions and of using general enough trace representations, while trying to curbing both representation-degeneration and catastrophic-forgetting risks. A detailed description of this algorithm is given in Sect. 5.3.

In order to address RG'', we devised an experimentation including, as a reference baseline, a method that trains the

[1] In order to apply (cross-domain) transfer-learning methods, one should find another business process, say P', such that: (i) P' is behaviorally similar to the business process under analysis, and (ii) far more outcome-labeled traces are available (or can be obtained easily) for P'. These assumptions rarely hold together in reality. On the other hand, t.b.o.k. there are no augmentation techniques specifically devised for process traces. Indeed, consolidated augmentation techniques were widely used on images, videos and speech data (e.g., for which it is easy to derive novel, surrogate, examples from a labeled image by modifying the colorization/rotation angle of objects in the image, while leaving the label of the image unchanged). However, defining augmentation techniques for categorical data (which play a key role in process logs) is a challenging research topic.

DDN-based outcome predictor in a fully-supervised way (as done by most state-of-the-art methods in the field (Kratsch et al. 2020; Teinemaa et al. 2019; Hinkka et al. 2018; Metzger et al. 2019)), in addition to SSOP − MTL, the pre-training-based method proposed in Folino et al. (2019), and a "feature-extraction" variant of it (also introduced in Folino et al. (2019)). This empirical study (involving a wider and more variegate collection of datasets than in Folino et al. (2019)) shows that the proposed method SSOP − MTL significantly outperforms both the supervised baseline and the above-mentioned semi-supervised competitors.

Organization: Section 2 introduces basic concepts and notation concerning both process log data and the outcome prediction problem, as well as some background on self-supervised learning. An overview of related work (in the areas of semi-supervised learning and predictive monitoring) is offered in Sect. 3, which also discusses the main points of novelty of our current proposal. Section 4 presents an abstract DNN learning framework that encompasses the semi-supervised discovery methods proposed in Folino et al. (2019) and in this work, which all rely on using next-activity prediction as an auxiliary (self-supervised) task. The training algorithms employed in these approaches are illustrated in Sect. 5. After discussing, in Sect. 6, the comparative experimental analysis performed over different real-life log data, we finally draw some concluding remarks and directions for future work in Sect. 7.

2 Background and Problem

2.1 Preliminaries: Events, Traces, Logs

A (process) log is a collection of traces, storing information on past executions of a business process. More specifically, each log trace represents the sequence of activity-related events happened during the execution of a process instance, and each of these events can have multiple data attributes.

Formally, let $\mathcal{E}$ and $\mathcal{T}$ be the universes of events and traces that could be generated by executing the process under analysis. Each event attribute A is a function $A : \mathcal{E} \rightarrow Dom(A)$ that assigns a value in the domain of A to any event in $\mathcal{E}$. For the sake of generality, let us assume that two lists of attributes are defined over $\mathcal{E}$: categorical attributes $A_1, \ldots, A_{m1}$ and numerical attributes $B_1, \ldots, B_{m2}$, for some $m1, m2 \in \mathbb{N}$. Three important kinds of information that are usually associated with any event e in real process logs are: *(i)* the activity executed in e, *(ii)* the resource/agent involved in the execution of the activity, and *(iii)* a timestamp that allows for ordering the events in a trace temporally.

We hereinafter assume the two former pieces of information to be encoded by two categorical event attributes $act(e)$ and $res(e)$, respectively, and the timestamp by a numerical attribute $time(e)$.

For any trace $\tau \in \mathcal{T}$, let $len(\tau)$ be the number of events in τ, and $\tau[i]$ be the i-th event of τ, for $i \in [1..len(\tau)]$. Moreover, let $\tau[:i]$ be the *prefix* (sub-)trace containing the first i events of a trace τ. For any complete trace τ, the prefixes $\tau[:i]$ represent partial enactments of the same process instance as τ.

For any $S \subseteq \mathcal{T}$, let $Prefs(S)$ be the set containing the prefixes of the traces in S, i.e., $Prefs(S) = \{\tau[:i] \mid \tau \in S \text{ and } 1 \leq i \leq len(\tau)\}$.

Finally, a *log* L is a finite subset of the trace universe $\mathcal{T}$.

2.2 General Formulation of the Problem Addressed: Discover an Outcome Prediction Model

Let $\mathcal{C}$ be a set of outcome-oriented classes defined over the process instances. The ultimate goal of outcome prediction methods is to predict the outcome class of any ongoing process instance at run-time, based on the (usually partial) trace currently available for the process instance itself. For the sake of presentation, let us assume that the (full or partial) traces have an associated (hidden) outcome class.

Let $\mu : \mathcal{T} \rightarrow \mathcal{C}$ be the unknown function that assigns a class label $\mu(\tau)$ to each $\tau \in Prefs(\mathcal{T})$.

Then, the general problem addressed in our work consists in discovering an *outcome prediction* model (or, more shortly, a *predictor*) $\tilde{\mu} : \mathcal{T} \rightarrow \mathbb{R}^{|\mathcal{C}|}$ that maps any trace $\tau \in Prefs(\mathcal{T})$ to a discrete probability distribution $\tilde{\mu}(\tau)$ over $\mathcal{C}$, such that the i-th element of $\tilde{\mu}(\tau)$ is an estimate of the probability that τ belongs to the i-th class in $\mathcal{C}$, and the elements in $\tilde{\mu}(\tau)$ sum up to 1.

In the literature, this problem, named hereinafter *outcome predictor discovery*, has been typically faced as a supervised learning problem over a given annotated log L, where each trace is associated with a class label representing its actual outcome (i.e., the value that μ takes on the trace). This annotated log is turned into a training set for learning the prediction model, where the prefixes of all τ in L, labeled with the same class as τ, are used as distinguished training examples. The resulting prediction model can be used to forecast the outcome $\mu(\tau')$ of whatever novel (unlabeled) trace $\tau' \in \mathcal{T}$.

In line with recent literature in the field (Kratsch et al. 2020; Mehdiyev et al. 2018; Tax et al. 2017; Evermann et al. 2017; Navarin et al. 2017; Hinkka et al. 2018; Teinemaa et al. 2018; Folino et al. 2019), we adopt a DNN-based approach to this discovery problem. However, in order to deal effectively with application scenarios featuring a relatively small number of labeled traces, we renounce resorting to pure supervised learning methods (like those underlying the great majority of the previous

approaches to the discovery of DNN-based outcome predictors (Mehdiyev et al. 2018; Tax et al. 2017; Evermann et al. 2017; Navarin et al. 2017; Hinkka et al. 2018; Teinemaa et al. 2018)) and try to also exploit non outcome-labeled traces, by employing the prediction of the next activity (for unfinished traces) as an auxiliary (self-supervised) learning task.

2.3 Self-Supervised Learning as a Form of Representation Learning

In general, self-supervised learning (Liu et al. 2021) is a way of learning good low-dimensional embeddings for complex/high-dimensional data (such as images, videos, multivariate sequences), which can be reused to perform other ("downstream") learning tasks (e.g., image/text/sequence classification). Recently, several self-supervised learning methods have been used as a valuable means for improving supervised learning systems (by extracting knowledge from the unlabeled data) when few labeled instances are available.

Differently from unsupervised learning, self-supervised learning relies on devising an auxiliary supervised task over the unlabeled data instances that generally consists in recovering information encoded via artificial target variables that: (a) can be derived from the data instances themselves automatically, and (b) must be predicted on the basis of other (i.e., incomplete, transformed, distorted or corrupted) parts of the data instances. For example, in Computer Vision, typical self-supervised tasks consist in rediscovering the rotation angle or colorization that were applied artificially to example images (in a preliminary step) (Liu et al. 2021). In natural language processing (NLP), instead, some popular pretext tasks employed for self-supervised representation learning are: Center/neighbor-words prediction (Mikolov et al. 2013), neighbor-sentences prediction (Kiros et al. 2015) and iteratively predicting the next word in a sentence, as a way of learning a *language model* (Bengio et al. 2003).[2] A self-supervised approach to learning representations for process traces was defined in Seeliger et al. (2021), which relies on training a recurrent neural net over a pretext task consisting in reconstructing one or multiple global attributes of a process instance for a completed trace, based on the full sequence of events in the trace. Interestingly, this self-supervised approach was shown in Seeliger et al. (2021) to yield representations that are more suitable for trace clustering, compared to those obtained with previous approaches to trace-representation learning.

The momentum gained by self-supervised learning methods (as an alternative to previous, unsupervised, representation-learning methods) mainly stems from the fact that they can leverage consolidated, efficient and robust supervised learning algorithms to address the auxiliary "pretext" task, but without needing manually-assigned data labels. However, since the ultimate goal of this task is to help recognize useful/transferable domain knowledge, it must be designed carefully. In particular, a good auxiliary task should enjoy the following two properties: (*P1*) it should be both general and complex enough, in order to oblige the DNN model to reason on semantical aspects of the data, and eventually learn general data representations that can be reused in other learning tasks; (*P2*) it should not be overly complex and demanding in terms of amount of training data, in order to prevent a DNN model trained on this tasks from incurring overfitting and from relying on useless data representations.

3 Related Work

3.1 Semi-Supervised Learning

Extending inductive-learning methods with the ability to apprehend effectively from few labeled examples (as regularly done by humans) is a hot challenging topic in machine learning. A consolidated solution approach consists in extracting hidden knowledge from unlabeled data, usually available in a larger number. This is the general aim of semi-supervised learning methods (see Van Engelen and Hoos (2020) and Ouali et al. (2020) for recent surveys on this topic), which can be divided into three major categories:

1. *Label-propagation* methods (Triguero et al. 2013), which alternate the training of one or multiple classifiers over labeled instances with the assignment of labels to unlabeled instances. This category includes self-training, pseudo-labeling, co-training, democratic co-learning, tri-training and associated variants (Triguero et al. 2013). In principle, label propagation methods can be instantiated with any classifier-learning method. However, they have found little

[2] Conceptually, a language model (LM) encodes a probability distribution over word sequences (taken from a language or text corpus). To learn a LM, one can train a neural network to predict the probability of each word, relatively to some linguistic context for the word itself. Usually, this neural-network classifier is trained, in an auto-regressive way, to predict the k-th word of any word sequence $s = s_1, \ldots, s_m$, based on the words that precede it in s – i.e., based on the prefix ending on the $(k-1)$-th word. Since minimizing the cross-entropy loss associated with this prediction task is the same as maximizing the data likelihood, the resulting model approximates a LM that represents the probability of all sequences $s = s_1, \ldots, s_m$ in the population of interest via chain-rule factorization, i.e., $P(s_1, \ldots, s_m) = P(s_1) \cdot P(s_2 \mid s_1) \cdot \ldots \cdot P(s_{m-1} \mid s_1, \ldots, s_{m-2}) \cdot P(s_m \mid s_1, \ldots, s_{m-1})$. Notably, the training of language models was used recently to address semi-supervised-learning problems (Qiu et al. 2020; Dai and Le 2015)

application in DL frameworks, since common procedures for learning a DNN model are too costly to be embedded in an iterative learn-and-classify scheme.

2. *Pre-training* methods, which exploit the unlabeled instances in a "task-agnostic" way (w.r.t. the target supervised task), in order to obtain meaningful data representations/embeddings that can be used as a basis for training the classifier on labeled examples. The embeddings learnt this way can be either used as such (as a sort of feature-extraction result) or "fine-tuned" over the given labeled data. Traditional ways of pre-training a DNN consist in learning some autoencoder/generative model (Van Engelen and Hoos 2020; Ouali et al. 2020). In the last few years, many successful pre-training approaches leveraged self-supervised learning methods (Liu et al. 2021; Qiu et al. 2020; Dai and Le 2015) to grasp knowledge from unlabeled data, as already mentioned in Sect. 2.3.
3. *Intrinsically semi-supervised learning* methods, which exploit both labeled and unlabeled data in a single training procedure, according to a multi-objective optimization strategy. In the case of DNN models, this means minimizing both a classification loss over the labeled examples and some auxiliary loss over the unlabeled ones. Most of the methods proposed in this field work in a "task-specific" way, with both losses defined in terms of the predicted class labels. Usually, the unsupervised loss term is a measure of inconsistency between the predictions obtained by using different perturbations of an unlabeled instance and/or different versions of the classifier. Examples of such methods are: ladder networks (Rasmus et al. 2015), Π-Model (Laine and Aila 2016), temporal ensembling (Laine and Aila 2016), mean teacher (Tarvainen and Valpola 2017) and virtual adversarial training (VAT) (Miyato et al. 2018). These methods were shown successful in image classification tasks, where the data instances have a continuous nature that allows for defining meaningful data perturbation schemes easily. However, devising approaches of this category for discrete data (like text and process traces) is an open research issue.

Conceptually, the discovery method $\texttt{SSOP} - \texttt{MTL}$ we are proposing here resembles the methods in the last category hereinabove, owing to its ability of exploiting labeled and unlabeled log data synergistically, based on minimizing two different loss functions related to the auxiliary and target tasks, respectively. However, two important features of our approach make it different from these methods: (i) it does not rely on generating perturbed versions of the labeled data (as it is done, instead, in Rasmus et al. (2015); Laine and Aila (2016); Tarvainen and Valpola (2017); Miyato et al. (2018)), and (ii) it automatically modifies the relative importance of the two losses, across the training, in order to allow for a gradual passage of bias between the two tasks.

3.2 Predictive (Process) Monitoring

Predictive process monitoring (shortly, predictive monitoring) (Maggi et al. 2014; Metzger et al. 2015) is an active line of research, which aims at supporting process monitoring frameworks with the ability of making per-instance predictions at run time. Great attention in this field has been attracted by the prediction of two categorical properties for a process instance: the next activity (Evermann et al. 2017; Tax et al. 2017; Navarin et al. 2017; Mehdiyev et al. 2018; Hinkka et al. 2018; Camargo et al. 2019; Lin et al. 2019; Pasquadibisceglie et al. 2019; Taymouri et al. 2020), and the final outcome (Teinemaa et al. 2018, 2019). Since both properties are usually categorical (in particular, possible outcome values are usually regarded as a predefined set of outcome classes), the solutions proposed for these prediction tasks look very similar technically, since they rely on discovering a classification model from a collection of labeled traces.

The usage of DNN-based methods (Kratsch et al. 2020; Mehdiyev et al. 2018; Tax et al. 2017; Evermann et al. 2017; Navarin et al. 2017; Hinkka et al. 2018; Teinemaa et al. 2018; Camargo et al. 2019; Lin et al. 2019; Pasquadibisceglie et al. 2019; Taymouri et al. 2020) became widespread in recent years, owing to their ability to grasp effective trace representations automatically. In Kratsch et al. (2020), it was empirically shown that DNN-based methods "generally outperform classical ML approaches when it comes to outcome-oriented predictive process monitoring". The superiority of DNN-based outcome predictors is quite neat against flexible business processes and/or logs with several event/trace attributes.

Most of the best-performing methods proposed so far (Mehdiyev et al. 2018; Tax et al. 2017; Evermann et al. 2017; Navarin et al. 2017; Hinkka et al. 2018; Teinemaa et al. 2018; Camargo et al. 2019; Lin et al. 2019) leverage LSTM (long short-term memory) architectures (Hochreiter and Schmidhuber 1997). For example, in the LSTM-based network proposed in Tax et al. (2017), for predict the next activity and its associated timestamp, each event e of a trace is encoded by concatenating numerical features derived from $time(e)$ with a one-hot representation of $act(e)$. A pretty similar architecture is proposed in Evermann et al. (2017) for next-activity prediction, which possibly employs ad hoc embedding layers to encode information on activities and executors. Several alternative LSTM-based architectures are proposed in Camargo et al. (2019) to predict the activity, timestamp and resource of

the next event, taking account for both categorical and numerical event attributes. Predicting all the categorical attributes of the next event is faced in Lin et al. (2019) by discovering an LSTM-based model including attention mechanisms for combining the outputs of different LTSM stacks (one per event attribute).

The LSTM-based outcome predictors evaluated in Teinemaa et al. (2018) and Kratsch et al. (2020) mainly look like an adaptation to the outcome prediction problem of the basic model proposed in Tax et al. (2017) (for next-activity prediction). Notably, in the extensive experimentations conducted in these works, this simple DNN architecture was shown to be very competitive against state-of-the-art ML-based approaches.

All of the methods described above rely on a supervised learning strategy, which makes them unsuitable for those outcome prediction contexts (e.g., involving the prediction of faults, frauds, customer satisfaction levels, etc.) where the outcome-class labels of many log traces are unknown or difficult to obtain. Clearly, this problem does not affect the traditional application scenarios where these methods are exploited to learn a DNN for predicting information (e.g., the next activity or some process/performance -related kinds of outcome) that is regularly stored for every completed process instance.

3.3 Semi-Supervised Approaches to the Prediction of Process Outcomes

To the best of our knowledge, the only attempt to exploit a semi-supervised learning strategy in a predictive-monitoring context has been made in Folino et al. (2019), which first stated the problem of discovering an outcome predictor in a label-scarcity setting. Essentially, the solution method proposed in Folino et al. (2019), named hereinafter $\texttt{SSOP}-\texttt{PT}$, consists in applying a pure pre-training procedure (see Sect. 3.1, category 2), using the prediction of the next activity of a trace as the auxiliary task. This method is illustrated in detail in Sect. 5.2, after presenting its underlying DNN architecture in Sect. 4.2.

As mentioned in Sect. 1, method $\texttt{SSOP}-\texttt{PT}$ may fail to ensure an optimal trade-off between two diverging objectives: *(o1)* conserving knowledge coming from the auxiliary task, in order to avoid spurious trace embeddings and overfitting; *(o2)* fitting well the target task, in order to eventually yield accurate outcome predictions. This limitation stems from the fact that the two tasks are used (rather independently) to lead the pre-training and fine-tuning phases, respectively, and exposes the method to two serious risks (described in more detail in Sect. 5.2): *(r1)* the embeddings learnt in the pre-training phase are messed up in the fine-tuning phase (a.k.a. catastrophic forgetting); and *(r2)* the pre-training phase leads the DNN to a region of the parameter space that overfits the auxiliary task, and prevents the DNN from fitting well the target task (a.k.a. representation degeneration).

Unfortunately, the empirical analysis in Folino et al. (2019) does not allow for assessing the robustness of $\texttt{SSOP}-\texttt{PT}$ to these risks in a wide enough range of application scenarios. The extensive experimentation discussed in Sect. 6 shows that $\texttt{SSOP}-\texttt{PT}$ is not always significantly better than a pure-supervised discovery – so filling the evidence gap in Folino et al. (2019) and enabling a deeper understanding of the behavior of $\texttt{SSOP}-\texttt{PT}$.

The discovery method proposed in this work, named hereinafter $\texttt{SSOP}-\texttt{MTL}$, addresses the same auxiliary task and target task as in Folino et al. (2019) but in a joint fashion, using a multi-task DNN architecture and a totally different training algorithm (which makes the weights of the two tasks vary dynamically). In our empirical study, this novel method is shown to surpass $\texttt{SSOP}-\texttt{PT}$, presumably owing to its ability to ensure a better balance between the goals *o1* and *o2* mentioned above – and hence a greater level of robustness to their associated risks *r1* and *r2*.

The next section presents a semi-supervised formulation of the discovery problem addressed in this work, with next-activity prediction used as an auxiliary (self-supervised) task. Based on this formulation, we will illustrate, in Sect. 5, the training algorithms implemented by $\texttt{SSOP}-\texttt{PT}$ and $\texttt{SSOP}-\texttt{MTL}$.

4 Semi-Supervised Outcome Prediction: High-Level DNN Architectures

4.1 The Chosen Auxiliary Task: Next-Activity Prediction

Moving from the basic framework proposed in Folino et al. (2019), we leverage the core idea of exploiting next-activity prediction as an auxiliary self-supervised learning task for outcome prediction. Essentially, this task consists in discovering a probabilistic neural-network classifier that, given any partial trace, is able to predict the activity that will be executed in the next step of the trace. In what follows, we first define this task formally, and then discuss the role that it is expected to play in our approach.

Formulation of the task: Let $a_1, \ldots, a_m$ be the activity labels in $Dom(act)$. Moreover, for each trace τ in $\mathcal{T}$, and every prefix $\tau' = \tau[:i]$ of τ (with $i \in [1..len(\tau)]$), let $nextAct(\tau')$ be a (unknown) function that maps τ' to a next-activity label as follows: $nextAct(\tau') = act(\tau[i+1])$ if $i < len(\tau)$, or $nextAct(\tau, i) = \texttt{EOS}$ otherwise, where $\texttt{EOS}$ is a special (dummy) "end-of-sequence" symbol.

Then, the auxiliary *next-activity prediction task* consists in guessing $nextAct(\tau)$, for any $\tau \in Prefs(\mathcal{T})$. As for outcomes prediction, the result of this task is a categorical probability distribution, specifically representing the probability that the next activity of τ is x, for each $x \in \{a_1, \ldots, a_m, \texttt{EOS}\}$.

Given a set L of completed traces, one can train a DNN classifier (as, e.g., in Evermann et al. (2017); Tax et al. (2017); Navarin et al. (2017); Mehdiyev et al. (2018); Hinkka et al. (2018); Camargo et al. (2019); Lin et al. (2019); Pasquadibisceglie et al. (2019); Taymouri et al. (2020)) to accomplish this (pretext) prediction task, providing it with labeled examples of the form $(\tau[:i], nextAct(\tau, i))$, such that $\tau \in L$ and $i \in [1..len(\tau)]$, where the next-activity labels play as class labels.

Suitability of this task for our problem setting: As suggested by the preliminary empirical study in Folino et al. (2019), we are confident in the fact that the above-described next-activity prediction task can play as a good auxiliary task in a semi-supervised outcome-prediction setting, since it enjoys both of the desired properties mentioned in the end of Sect. 2.3.

Specifically, as concerns property *P1*, this task is expected to force the DNN model to grasp some understanding of the business process that produced the log traces, and to reason in terms of general (non incidental) properties of them (e.g., linked to business rules, typical execution schemes, modi operandi, or exception patterns). Such a semantic/generalization bias will keep the DNN model away from overfitting, and make it more robust to the risk of relying on spurious prediction patterns. However, since, in general, this auxiliary task is not generally ensured to have some strong correlation to the outcome-prediction task, we will devise a discovery strategy that harmonizes the two tasks in a way that balances the need of guessing the outcome classes on the training examples, on one hand, and that of avoiding spurious trace embeddings, on the other hand.

Moreover, next-activity prediction is not too complex and data hungry (property *P2*), for it was shown empirically accurate DNN-based models can be found for this task using typical (or even very small) amounts of log traces (Käppel et al. 2021). In our opinion, this property is hardly exhibited by two, more complex, tasks considered in the process-mining literature: (i) predicting the cycle/remaining time of a process instance, which entails learning a real-valued function, and (ii) predicting multiple attributes of the next event.

In principle, one could regard process traces as sequences of activity labels, and simply define the auxiliary task as the discovery of a model that predicts the next activity of a process instance, based on the sequence of activities that it has been producing so far. Since this task coincides with the discovery of a *language model* (where the language consists of all the possible activity sequences that can be generated by the process under analysis), this makes it possible to reuse the large body of solutions developed in NLP for the discovery of language models (Qiu et al. 2020; Dai and Le 2015). However, besides overlooking the peculiarity of process traces and the differences between processes and languages (in terms, e.g., of vocabulary size, cardinality, long-term dependencies), we discard such a simplified formulation of the auxiliary task, because it suffers from a serious drawback: it focuses only on control-flow aspects, while totally disregarding the "multi-modal" information conveyed by other event attributes (i.e., attributes that do not represent activity labels), which can be very useful for the activity-prediction and outcome-prediction tasks (and for deriving informative trace representations) – and, in fact, all state-of-the-art methods for the discovery of deep outcome predictors exploit both these attributes (see Kratsch et al. (2020)).

4.2 DNN Architectures

All the DNN models considered in our framework follow the abstract encoder-predictor architectures sketched in Fig. 1, which use the same structure for their encoder sub-net f. This sub-net is meant to take an event sequence of the form $\tau = e_1, \ldots, e_n$ as input, and to map it onto a vectorial representation $z = f(\tau)$. The ultimate goal of this subnet is to extract general trace embedding, which can be suitably transferred/shared between the auxiliary task and the target one, so that the latter can benefit from the additional supervision coming from the auxiliary task.

Different structures can be adopted for the predictor sub-net, using two basic kinds of blocks: (i) the activity-oriented predictor block g^A, which returns a discrete probability distribution over the set of activity labels augmented with the special end-of-sequence (EOS) label; and (ii) the outcome-oriented predictor block g^O, which returns a discrete probability distribution over the set $\mathcal{C}$ of outcome classes.

More specifically, we consider three different DNN architectures (which are at the basis of the semi-supervised outcome prediction approaches considered in this work):

- M^A, consisting of the encoder sub-net f surmounted by the predictor block g^A, which is meant to only support the auxiliary next-activity prediction task by returning, for any given trace τ, the probabilities $P(a \mid \tau)$ for each $a \in Dom(act) \cup \{\texttt{EOS}\}$;
- M^O, featuring the encoder sub-net f and the predictor block g^O, which is meant to only support the target outcome prediction task by returning, for any given

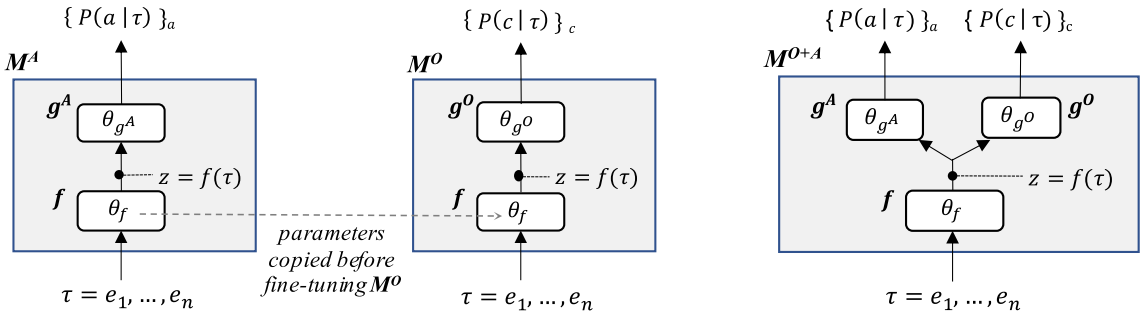


Fig. 1 Abstract DNN architectures adopted for the self-supervised discovery of an outcome predictor. Models M^A (left) and M^O (middle) are used in the pre-training-based methods SSOP − PT and Base − FE introduced in Folino et al. (2019), whereas model M^{O+A} (right) is meant to support the multi-task-learning algorithm SSOP − MTL. See Sect. 5 for more details on the algorithms

trace τ, the probabilities $P(c \mid \tau)$ for each outcome class $c \in \mathcal{C}$; and

– M^{O+A}, featuring both predictor blocks g^O and g^A on top of f, which accomplishes the outcome prediction and activity prediction tasks jointly, by returning, for any given trace τ, both probabilities $P(a \mid \tau)$ and $P(c \mid \tau)$, for each $a \in Dom(act) \cup \{\texttt{EOS}\}$ and each $c \in \mathcal{C}$.

Clearly, in all of these models, the final prediction for any given trace τ is always made on the basis of the representation $z = f(\tau)$ provided by the encoder sub-net.

In principle, the abstract sub-nets f, g^A and g^O could be instantiated by using different alternative DNN architectures; this makes the discovery framework described here inherently parametric with respect to the structure of these sub-nets. However, for the sake of concreteness and of comparison, in our experimental analysis we made the following choices, as done in Folino et al. (2019) and in Teinemaa et al. (2018): (i) both predictor blocks g^A and g^O consist each of a single dense (feed-forward) layer that returns a vector of probabilities, containing as many components as the EOS-augmented activity labels and the outcome classes, respectively; (ii) the encoder block f is a stack of LSTM layers (plus an event embedding layer), as discussed in more detail in Sect. 6.1 and 6.3. Following previous works in the literature, the neural units of g^O (resp., g^A) are equipped with a *sigmoid* (resp., *softmax*) activation function. Implementing and testing alternative choices for the internal architecture of f is left to future work, as discussed in Sect. 7.

Models M^A and M^O were already used in the two discovery methods defined in Folino et al. (2019), which are presented in Sect. 5.2. The hybrid architecture M^{O+A} is at the basis of the semi-supervised discovery method SSOP − MTL that we are proposing in this work, and which is explained in detail in Sect. 5.3.

Before delving into the technical details of these methods, in the following section, we first introduce a semi-supervised formulation of the discovery problem addressed by all these methods.

5 Semi-Supervised Discovery: Existing Methods and Our Proposal

5.1 Notation: Semi-Supervised Formulation of the Discovery Problem

In general, the source of information for training DNN models of the forms described in Sect. 4.2 is a given log L, which consists of multiple fully-unfolded process traces, each of which is either equipped with an outcome-class label (labeled traces) or not (unlabeled traces).

In order to learn a DNN model in a self-supervised way, a log L must be converted into a training set $\mathcal{D}$ of annotated event sequences that represent all the prefixes of the traces in L (and hence all the partial executions of the corresponding process instances). Each of these sequences is annotated with two labels, representing the ground-truth next activity and outcome class, respectively.

Formally, let us denote each training instance in $\mathcal{D}$ as a triple $d = \langle trace, nextAct, out \rangle$, where $d.trace \in Prefs(L)$ is a (partial) trace, while $d.nextAct \in Dom(act) \cup \{\texttt{EOS}\}$ is the next-activity label associated with $d.trace$ (i.e., $d.nextAct = nextAct(d.trace)$) and $d.out \in \mathcal{C} \cup \{\perp\}$ is the ground-truth outcome class of $d.trace$ (i.e., $d.out = \mu(trace)$) if available, or $\perp$ otherwise.

In what follows, we first describe, in Sect. 5.2, two existing semi-supervised methods for training a DNN-based outcome predictor on such a dataset (namely, the pre-training approach defined in Folino et al. (2019), and a feature-extraction variant of it), while discussing some major drawbacks of these methods. In Sect. 5.3, we then illustrate a novel algorithm, named SSOP − MTL, that implements the specific multi-task-learning solution we are proposing in this work.

5.2 Two Existing Semi-Supervised Methods and Their Points of Weaknesses

5.2.1 A pre-training approach: method SSOP − PT

Provided with an annotated training set $\mathcal{D}$ of the form described above, the method proposed in Folino et al. (2019), and named hereinafter SSOP − PT, performs the following two computation phases in a sequence:

1. *Unsupervised pre-training:* First, a next-activity predictor of the form M^A is trained on $\mathcal{D}$, independently of the outcome-class labels. To this end, a (locally) optimal configuration of the network parameters $\theta^A \equiv (\theta_f, \theta_{g^A})$ is found by using a standard mini-batch gradient descent procedure (see Sect. 6 for details) for optimizing the following loss function:

$$\mathcal{L}^A(\theta^A) = -\frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \log P\big(d.nextAct \mid d.trace;\ \theta^A\big) \tag{1}$$

   where, for each trace τ and each activity $a \in Dom(act) \cup \{\texttt{EOS}\}$, $P(a \mid \tau;\ \theta^A)$ is the probability value that M^A returns (based on the current configuration of its internal parameters, i.e., DNN weights, θ^A) for a, when taking τ as input.
2. *Knowledge transfer and supervised fine tuning:* The optimal values found for the parameters θ_f, at the end of the previous step, are used to initialize the encoder sub-net of an outcome prediction model M^O, whereas the other parameters of M^O are set randomly. Then, the pre-trained model M^O is adapted to the outcome-prediction task by training it over the outcome-labeled traces. Specifically, let $\mathcal{D}^{lab}$ denote the subset of training instances that have an associated outcome label, i.e., $\mathcal{D}^{lab} = \{d \in \mathcal{D} \mid d.out \neq \perp\}$. Then, an optimal configuration of the parameters $\theta^O \equiv (\theta_f, \theta_{g^O})$ of M^O is found by minimizing the following loss function (still through mini-batch gradient descent):

$$\mathcal{L}^O(\theta^O) = -\frac{1}{|\mathcal{D}^{lab}|}\sum_{d\in\mathcal{D}^{lab}} \log P\big(d.out \mid d.trace;\ \theta^O\big) \tag{2}$$

where, for each trace τ and each outcome class $c \in \mathcal{C}$, $P(c \mid \tau;\ \theta^O)$ is the probability value that M^O returns (based on the current state of its parameters θ^O) for c, when taking τ as input.

Weaknesses of SSOP − PT: The pre-training strategy underlying SSOP − PT is exposed to two main kinds of risks, which may undermine the quality of the discovery outcome predictors:

- *Catastrophic forgetting* (Liu et al. 2021; French 1999): The knowledge gained by M^A in the pre-training phase and transferred to the lower layers of M^O vanishes during the fine-tuning of M^O on the outcome-labeled data, especially if this fine-tuning phase involves many training steps. In our setting, this means that the data transformation function learned by the encoder sub-net through next-activity prediction (and presumed to yield compact semantic representations of the traces) is rearranged completely in the fine-tuning phase. This prevents the model being trained to benefit from the process-aware understanding abilities acquired in the pre-training phase, which could turn very helpful when the labeled data are available for the fine-tuning are few (and the risk of overfitting them is high).
- *Representation degeneration (a.k.a. embedding degeneration)* (Liu et al. 2021): In the pre-training phase M^A overfits the auxiliary task, with the encoder sub-net focusing on data representations that undermine the ability of M^O to eventually adapt well to the main (outcome prediction) task.

As confirmed indirectly by experimental findings presented in Sect. 6, these two issues can reduce the advantage of using a self-supervised learning strategy, independently of how much the auxiliary task adopted is correlated to the target one. The hybrid architecture M^{O+A} above aims at mitigating both these risks (and hence better balance between knowledge conservation and adaptation to the target task), by enabling a more synergistic interplay between the two learning tasks.

5.2.2 A "Feature-Extraction" Variant of SSOP − PT: Method Base − FE

In Folino et al. (2019), a variant of the pre-training method SSOP − PT was defined where the internal parameters of the encoder subnet f are kept fixed ("frozen"), during the fine-tuning phase, to the value that is found at the end of the pre-training phase.

This variant, named Base − FE, simply consists in reusing the pre-trained encoder subnet f as a (non-trainable) "feature-extraction" function returning a dense representation for any trace, and training the predictor block g^O (acting as a logistic-regression classifier) on such trace representations. In other words, Base − FE (suffix *FE* here stands for Feature Extraction) founds on the idea of just exploiting activity prediction as a preliminary representation-learning task, and then reusing the trace representations so obtained as input features for outcome prediction.

This method can be regarded as a drastic solution for preventing catastrophic-forgetting phenomena, In principle, such an extreme approach could perform better than SSOP − PT when the latter incurs severe phenomena of this kind. This is the reason why this sort of "truncated" version of SSOP − PT will be considered in the experimental analysis of Sect. 6 (as done in Folino et al. (2019)).

However, as there is no way for Base − FE to adapt its internal trace representations to the outcome prediction task, the risk of representation degeneration exacerbates for this method. This prevent Base − FE from being an optimal solution for our problem setting – as confirmed by the empirical study in Sect. 6.

5.3 A Novel Semi-Supervised Method for Outcome Prediction: SSOP − MTL

It was shown empirically in previous works (see, e.g., Dai and Le (2015) in an NLP context) that pre-training methods can be improved (in terms of both convergence and generalization power) if the auxiliary task adopted for representation learning (in some pre-training step) is also taken into account in the fine-tuning phase. In order to enjoy this beneficial effect, we propose to adopt a multi-task DNN model of the form M^{O+A} presented in Sect. 4.2, complementing the outcome predictor g^O with a next-activity predictor g^A.

However, we do not want to use such a multi-task-learning approach just in the fine-tuning phase of a pre-training-based strategy like the one adopted by SSOP − PT. By contrast, we propose to train the model M^{O+A} from scratch, according to a multi-objective training strategy, which pursues the activity-prediction and outcome prediction in a joint manner, as a concrete way of addressing the main research goal RG' stated in Sect. 1.3.

Specifically, provided with a training set $\mathcal{D}$ of the form defined in Sect. 5.1, the discovery algorithm SSOP − MTL we are proposing here tries to minimize (iteratively, on different batches of data extracted from $\mathcal{D}$) the following loss function (which is a linear combination of those in Eqs. 1 and 2):

$$\mathcal{L}^{O+A}(\theta^A, \theta^O) = \lambda \cdot \mathcal{L}^A(\theta^A) + (1-\lambda) \cdot \mathcal{L}^O(\theta^O). \tag{3}$$

Clearly, factor λ in Eq. 3 controls the level of bias towards learning trace representations that really serve the auxiliary task (the higher λ, the stronger the bias). In simple multi-target learning frameworks, the analyst is in charge of setting a suitable value for λ, which is kept fixed across all the training process. However, such an approach hardly leads to the discovery of accurate outcome predictors when few outcome-labeled example traces are available, even assuming that the optimal value of λ is guessed.

Thus, in the final part of the training procedure, we propose to pay more attention to the errors made in the prediction of outcomes than in the prediction of next-activity labels; in our vision, the latter labels, indeed, are meant to offer a complementary source of supervision in initial/intermediate steps of the training process, to guide the encoder sub-net towards general trace embeddings (before eventually adapting these embeddings to the outcome-prediction task). To this end, we refine the training (mini-batch gradient-descent) procedure of our discovery algorithm SSOP − MTL in a way that the weight factor λ in the loss function of Eq. 3 varies dynamically, depending on the counter of training epochs performed. Two alternative schemes for implementing such a dynamical weighting of the tasks are proposed below.

Alternative loss-weighting schemes: hyperparameter $\lambda - sched$ Let N be the number of training epochs, each of which entails a complete optimization round over all the instances of $\mathcal{D}$, grouped in mini-batches. For the sake of both convergence and efficiency, the value of λ is only changed every $\lceil N/Q \rceil$ epochs in a piecewise way, while keeping it fixed till the next change point.

The value of λ on each training epoch $i \in [1..N]$ is computed by applying one of the two functions defined below (and sketched pictorially in Fig. 2) to the index $j = i \bmod \lceil N/Q \rceil$:

$$\lambda_{2\_\mathtt{phase}}(j) = \frac{\exp(j - N/2)}{1 + \exp(j - N/2)} \tag{4}$$

$$\lambda_{3\_\mathtt{phase}}(j) = \begin{cases} \dfrac{\exp(j - N/3)}{1 + \exp(j - N/3)} & \text{if } j \geq N/2 \\ \dfrac{\exp(j - 2N/3)}{1 + \exp(j - 2N/3)} & \text{otherwise} \end{cases} \tag{5}$$

where mod stands for the *modulus* operator, and $Q = 10$ was set in all the tests described in Sect. 6.

Specifically, algorithm SSOP − MTL is equipped with an ad hoc hyperparameter, named $\lambda - sched$ and ranging over $\{2\_\mathtt{phase}, 3\_\mathtt{phase}\}$, which allows for choosing among functions $\lambda_{2\_\mathtt{phase}}$ and $\lambda_{3\_\mathtt{phase}}$.

When choosing function $\lambda_{2\_\mathtt{phase}}$ (by setting $\lambda - sched = 2\_\mathtt{phase}$), model M^{O+A} is initially trained on the basis of the auxiliary task only, the loss of which is given the highest weight possible ($\lambda = 1$) at the first epoch (while setting the loss of the target task to 0). In the subsequent epochs, the weight λ decreases towards 0 according to a (upside-down) sigmoidal shape. This allows the algorithm to simulate a sort of gradual shift from a (self-supervised) pre-training mode to a (pure supervised) fine-tuning mode.

Our test results (cf. Sect. 6) confirmed that this solution often outperforms the pre-training approach of algorithm SSOP − PT, presumably owing to its ability of reducing the

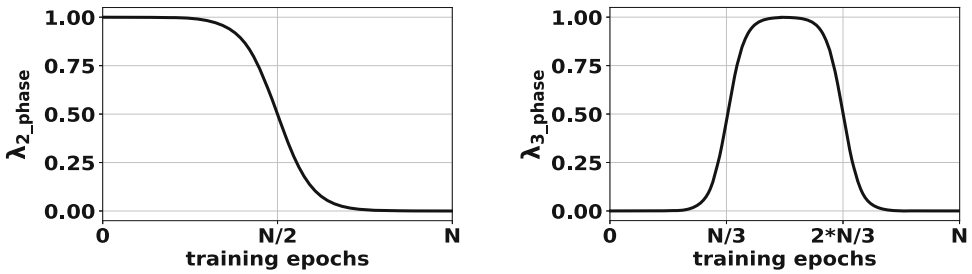


Fig. 2 Functions considered for dynamically changing the weight λ of the auxiliary loss over the training epochs: λ_{2_phase} (left) and λ_{3_phase} (right). These functions are chosen when setting $\lambda - sched = \texttt{2\_phase}$ and $\lambda - sched = \texttt{3\_phase}$, respectively

risks of *catastrophic forgetting* and of *representation degeneration* mentioned in Sect. 5.2. However, this solution was not very effective in some of the test scenarios, where we suspect that the encoder sub-net f overfitted the auxiliary task in the former half of the training process, and got trapped in a region of the parameter space from which it could not move to a point that suits the outcome prediction task well.

The "hat-shaped" function λ_{3_phase} (chosen when setting $\lambda - sched = \texttt{3\_phase}$) is meant to prevent this behavior by simulating a gradual shift of the optimization objective across three consecutive phases, which are guided, respectively, by: (i) the outcome prediction task (low values of λ), (ii) the auxiliary task (high values of λ), and (iii) the outcome prediction task again (low values of λ gradually tending towards 0, so that the model is eventually fine-tuned on the outcome-labeled data).

6 Experimental Analysis

6.1 Analysis Scope and Objectives: Discovery Methods and Experimental Hypotheses

6.1.1 *Methods Tested:* SSOP − MTL, *Semi-Supervised Competitors and a Fully-Supervised Baseline*

In order to fully address the research goal RG'' stated in Sect. 1.3, we experimentally compared the approach proposed here with several alternative ones, over different process logs, paying special attention to application settings where the ground-truth outcome classes are known for a relatively small fraction of the training traces.

Specifically, in this empirical study, the following semi-supervised approaches to the discovery of an outcome predictor were considered:

- the algorithm SSOP − MTL defined in Sect. 5.3, which encodes the method proposed in this work;
- the algorithm SSOP − PT described in Sect. 5.2, which encodes the only existing semi-supervised solution to the problem (proposed in Folino et al. (2019)):
- the "feature-extraction" variant Base − FE (also described in Sect. 5.2 and originally defined in Folino et al. (2019)) of SSOP − PT.
- a fully-supervised *baseline* method, named Base − S, which consists in training a DNN-based outcome predictor of type M^O in a supervised way, by minimizing the loss $\mathcal{L}^O$ only over the outcome-labeled instances at hand.

It is worth noting that the latter (baseline) method is meant to play as an archetype for the broad class of supervised approaches that dominate the field of outcome prediction and, more generally, of predictive process monitoring (Teinemaa et al. 2018, 2019; Kratsch et al. 2020; Hinkka et al. 2018).

All these methods were implemented by using the same LSTM-based instantiation of the encoder sub-net f of the DNN models, namely M^A, M^O and M^{O+A} (cf. Sect. 4.2), employed by the methods. Specifically, we resorted to nearly the same LSTM architecture (apart from an additional embedding layer) as in Teinemaa et al. (2018) and Kratsch et al. (2020) – which was shown to achieve compelling accuracy in the prediction of outcomes, compared to different ways of combining manually-extracted features and state-of-the-art ML methods.

All the discovery methods and the evaluation procedure were implemented in *Python 3.7.7*, using the popular *Keras* and *TensorFlow 2.0* deep learning APIs.[3]

6.1.2 *Experimental Hypotheses*

The experiments were aimed at assessing two different hypotheses:

[3] The source code (re-using some log parsing and preprocessing scripts made available in Teinemaa et al. (2018)), can be accessed by reviewers at the following link: https://tinyurl.com/em7h9fv2.

– *HP': At least one of the two existing semi-supervised methods* SSOP − PT *and* Base − FE *performs significantly better than the supervised baseline over the logs, when the percentage of labeled training data available is "small"..*[4]
– *HP": The semi-supervised method* SSOP − MTL *proposed here performs significantly better than the competitors* SSOP − PT *and* Base − FE *and than the supervised baseline* Base − S, *when the percentage of labeled training data available is "small".* [4]

We pinpoint that, owing to catastrophic forgetting and representation degeneration issues, we doubted whether hypothesis *HP'* held in reality, whereas we were quite confident in the validity of hypothesis *HP"*. Both these feelings were confirmed by the experimental results, as discussed in Sect. 6.4.

6.2 Testbed: Datasets and Evaluation Procedure

6.2.1 Datasets: Logs, Outcome classes, Data Pre-Processing

For the sake of comparison and reproducibility, we reused five of the outcome-annotated datasets employed in Teinemaa et al. (2018), which were all derived from real-life process logs (available at https://data.4tu.nl).

Three of the datasets, named hereinafter L_1^a, L_1^c and L_1^d, were extracted from log *BPIC 2012*, generated by a loan-application process of a Dutch financial institute. These datasets consist all of the same traces, but differ in the assignment of the outcome labels. In all cases, each trace was assigned a label based on the final status (namely, i.e., *accepted*, *canceled* or *declined*) of the loan application that originated the trace, according to a binary classification scheme (*positive* outcome vs *negative* outcome) (Teinemaa et al. 2018, 2019). Precisely, each trace in L_1^a (resp. L_1^c, L_1^d) was labeled as *positive* iff the final status of the loan application was accepted (resp. canceled, declined), and as *negative* otherwise.

The other two datasets, renamed here L_2 and L_3, were derived from the logs *Hospital billing* and *Road traffic fines*, respectively. Each trace in L_2, each concerning the handling of a case in the billing system of a hospital, is associated with an outcome class indicating whether the case was actually reopened (*positive*) or not (*negative*). By contrast, each trace in L_3, storing the history of a road-traffic fine (in the database of an Italian local police force), was assigned a binary (positive vs negative) outcome class distinguishing whether the fine was either repaid in full or not (and then sent for credit collection).

Table 1 shows the following kinds of summary information on each of these datasets: the number of traces/events, the average/maximum trace length, the attributes associated with log traces/events, and the relative frequency of the positive outcome class. Further details on the datasets can be found in Teinemaa et al. (2018).

Pre-processing log data and turning them into tensors Before converting each of the datasets into a set of trace prefixes, we cut every trace τ in it by removing the suffix of τ starting with the first event of τ disclosing the outcome class of τ, in order to avoid favorable prediction biases (as done in Teinemaa et al. (2018)).

In order to put the training/test data into a tensorial form, we had to choose a fixed length (i.e., a fixed number of events) for all the trace prefixes. For the sake of comparison with Teinemaa et al. (2018), we fixed this length to 40 (resp. 8, 10) for all the versions of log L_1 (resp. L_2, L_3), and then truncated every (complete) log trace longer than 40 (resp. 8, 10) events, before extracting its prefixes. The resulting trace prefixes were left-padded with zeros when containing less than the chosen trace length.

6.2.2 Test Procedure, Evaluation Metrics and Statistical Test

Test procedure – simulating different label-scarce scenarios (via label %): On each dataset, we tested all the methods under analysis according to an 80-20 temporal hold-out scheme (as in Teinemaa et al. (2018, 2019)), using the older 80% of the data instances for training and the remaining ones for test. This splitting prevents unwanted forms of future-information leakage (Sheridan 2013). 20% of the training instances were then used, as a validation set, to eventually select the best DNN model among those found in different training epochs. To this end we implemented an ad hoc temporal splitting procedure, ensuring the resulting validation set to have a balanced class distribution and contain the most recent examples for each class.

To test the sensitiveness of any discovery method to the scarcity of labeled data, we simulated various scenarios where only the outcome labels of a fraction *label*% of the training instances can be used to train the outcome predictors, while the remaining instances are regarded as unlabeled – in practice, we masked the labels of the latter instances by replacing them with the dummy class $\perp$. In this simulated application scenario, the supervised baseline method Base − S could only exploit the information provided by the fraction *label*% of training instances that were

[4] Before testing these hypotheses, we tried to quantify the notion of "small labeled data" pragmatically. To this end we performed a series of tests with the fully-supervised baseline alone, aimed at identifying a subset of values for this percentage that represented a critical application scenario for this method. Details on these preliminary tests are given in Sect. 6.2.2.

Table 1 Descriptive statistics and trace/event attributes concerning the datasets used in the experiments

Dataset	#Traces	#Events	Positive label ratio	Avg len.	Max len.	Categorical attributes (cardinality)	Numerical attributes
L_1^a	4685	155,783	0.48	35	175	Activity (36), resource (63)	Hour, weekday, month, event_nr, timesincemidnight, timesincelastevent, timesincecasestart, open_cases, AMOUNT_REQ*
L_1^c	4685	155,783	0.35	35	175	Activity (36), resource (63)	Hour, weekday, month, event_nr, timesincemidnight, timesincelastevent, timesincecasestart, open_cases, AMOUNT_REQ*
L_1^d	4685	155,783	0.17	35	175	Activity (36), resource (63)	Hour, weekday, month, event_nr, timesincemidnight, timesincelastevent, timesincecasestart, open_cases, AMOUNT_REQ*
L_2	77,525	404,721	0.05	6	217	Activity (17), resource (668), actOrange (3), actRed (3), blocked (2), caseType (9), diagnosis (1,016), flagC (3), flagD (2), msgCode (11), msgType (3), state (10), version (8), speciality* (23)	Hour, weekday, month, event_nr, timesincemidnight, timesincelastevent, timesincecasestart, open_cases, msgCount
L_3	129,615	460,462	0.46	4	20	Activity (10), resource (270), notificationType (3), lastSent (4), dismissal (8), article* (85), vehicleClass* (4)	Hour, weekday timesincelastevent, timesincecasestart, timesincemidnight, month, expense, event_nr, open_cases, amount*, points*

The cardinality of each categorial attribute is reported within round brackets. Global attributes of the traces are annotated with symbol *, in order to allow for easily distinguishing them from event attributes

being considered as labeled, while disregarding all other training instances. By contrast, the semi-supervised methods $\texttt{SSOP}-\texttt{MTL}$, $\texttt{SSOP}-\texttt{PT}$ and $\texttt{Base}-\texttt{FE}$ could take advantage of all the training instances, including those with masked outcome labels.

Specifically, we made *label*% vary in $\{2.5\%, 5\%, 10\%, 20\%, 40\%\}$ and, for each value in this set, we choose a corresponding number of instances via standard stratified sampling. Please note that, for some of the analyses described in the following, we also considered the ideal scenario *label*$\% = 100\%$, where all the instances in the training set have an associated (visible) outcome label.

Evaluation metrics: We used two standard metrics to assess the quality of any discovered outcome predictor: *(i) ACC* (accuracy), returning the percentage of test prefixes classified correctly; *(ii) AUC*, which measures the area under the *ROC Curve*. Note that the threshold-independent metric *AUC* is more informative than *ACC*, and more reliable over imbalanced datasets, like L_1^c, L_1^d and, especially, L_2. This is the reason why, hereinafter, we will consider *AUC* as the first choice for comparative analyses, and *ACC* as a second-level support for comparing methods achieving very similar *AUC* scores.

Devising a reference label-scarce scenario – identifying "small" percentages of labeled data: Prior to conducting our comparative analysis, we had to quantify the notion of "small" percentage of labeled training data. To this end, we tried to identify a critical application scenario for the fully-supervised method $\texttt{Base}-\texttt{S}$, where it performed significantly worse than in the ideal situation where all the training traces have an associated outcome-class label. As described in more detail in Appendix A (available online via http://link.springer.com), we found that this scenario can be made correspond to the case where the information on outcome labels is available for a tenth of the training traces at most (i.e., when *label*$\% \in \{2.5\%, 5\%, 10\%\}$) – and the remaining traces are considered as they were unlabeled. This challenging scenario will be simply denoted hereinafter as *label*$\% \leq 10\%$. It is worth noting that the performance degradation of $\texttt{Base}-\texttt{S}$ in this scenario, compared to the ideal one where *label*$\% = 100\%$, was assessed to be statistically significant by applying the Friedman-Nemenyi procedure (described in the following) to the two series of *AUC* scores obtained, across the datasets, in these two usage scenarios. By the way, this result confirms our expectation that a fully supervised DNN-based approach to the discovery of an outcome

predictor is not suitable when having a small number of labeled traces.

Statistical test of significance: To assess the significance of the differences observed between methods in a given series of tests (e.g., for different percentages of labeled data, or over different logs), we resorted to the *Friedman-Nemenyi* test (Demsar 2006; Garcia and Herrera 2009), a statistical procedure that has been widely used for evaluating classifiers. In this procedure, a (non-parametric) Friedman test is first used to possibly reject the null hypothesis H_0 that the populations of results (computed at different percentages of labeled data over all logs) produced by the different methods have the same mean, so that they can be considered as statistically different. Whenever this test rejects H_0, a Nemenyi test with a significance level of 0.05 is used (as post-hoc test) to detect the pairs of significantly different methods: if a pair of methods is assigned a p-value under 0.05, these methods are eventually deemed as significantly different from a statistical viewpoint.

6.3 Configuring the Structure of the DNN Models and the Hyperparameters

Structural design choices: As mentioned above, we restricted all the analyzed methods to use a DNN model where the encoder sub-net f (sketched in Fig. 3) contains a stack of D LSTM layers and eventually returns the state of the rightmost LSTM cell in the top-most layer, as done in Teinemaa et al. (2018). All these layers have the same number of units (indicating the dimensionality of their output space), and are equipped with standard batch-normalization and internal dropout mechanisms. This reflects our desire of defining the concrete structure of the DNNs in a way allowing the baseline method $\texttt{Base}-\texttt{S}$ to play as a good representative for existing fully-supervised approaches to DNN-based outcome prediction.

The good performances shown in Teinemaa et al. (2018); Kratsch et al. (2020) by LSTM-based predictors made them natural candidates for our test setting. However, since we also had to simulate stressing discovery scenarios affected by a scarcity of labeled examples, we decided to extend the encoder sub-net with an ad hoc *event embedding* layer, in order to reduce the risk of obtaining an overfitting DNN when using few labeled data. For any given event e_i, the event-embedding layer simply returns the concatenation of the following per-attribute representations: *(i)* the (scalar) value $B_j(e_i)$, for each numerical event attribute B_j; *(ii)* either the one-hot representation of $A_j(e_i)$ or a dense representation of $A_j(e_i)$, obtained by using a trainable embedding matrix, for each categorical attribute A_j.

Setting of the hyperparameters: In setting the hyperparameters (of the DNN models and training procedures), we did not seek an optimal configurations for each run of a method, but only a configuration of the baseline method $\texttt{Base}-\texttt{S}$ meeting two requirements: (r1) in a classic learning scenario $label\% = 100\%$, $\texttt{Base}-\texttt{S}$ performs equivalently to, or better than, the LSTM-based method in Teinemaa et al. (2018) and Kratsch et al. (2020; r2) $\texttt{Base}-\texttt{S}$ is as simple as possible in terms of the number of parameters to be optimized, in order to reduce the risk of overfitting small amounts of outcome-labeled data. A detailed description of the concrete hyperparameter settings used in the tests can be found in Appendix B.

6.4 Test Results

6.4.1 *Assessing Hypotheses HP′ and HP″ in the Label-Scarce Scenario ($label\% \leq 10\%$)*

To assess the hypotheses HP' and HP'', we tested the semi-supervised discovery methods $\texttt{SSOP}-\texttt{MTL}$, $\texttt{SSOP}-\texttt{PT}$ and $\texttt{Base}-\texttt{FE}$ in the critical operating scenario $label\% \leq 10\%$ we had previously identified for the supervised baseline $\texttt{Base}-\texttt{S}$ (see Sect. 6.2.2, paragraph *Devising a reference*

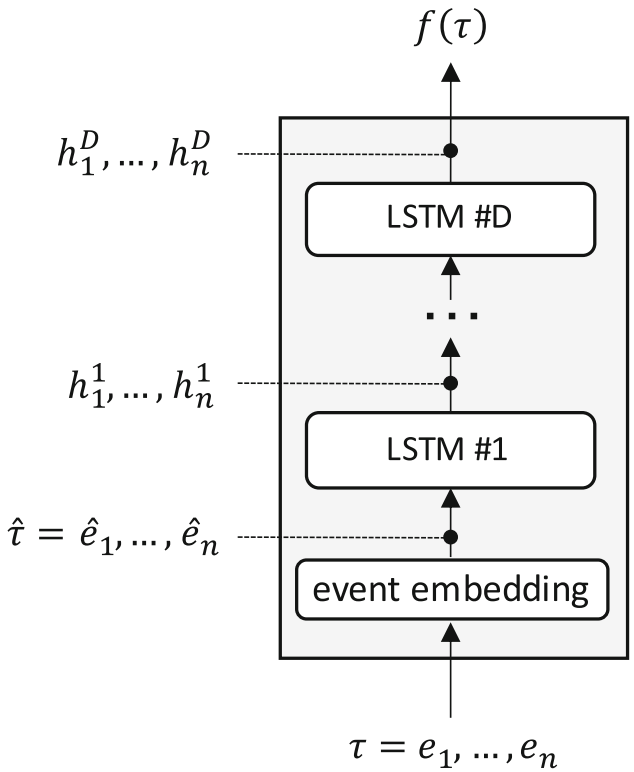


Fig. 3 Concrete architecture chosen for the encoder sub-net f of the DNN predictors (M^A, M^O, M^{O+A}) employed in the discovery methods evaluated in the experimentation

Table 2 *ACC* and *AUC* scores obtained by the semi-supervised discovery methods SSOP − MTL and SSOP − PT (Folino et al. 2019) and the baselines (i.e., Base − FE, and Base − S) when using small percentages of labeled data (i.e., *label*% ∈ {2.5%, 5%, 10%})

Dataset	Method	ACC	AUC
L_1^a	SSOP − MTL	**0.615**	**0.687**
	SSOP − PT (Folino et al. 2019)	0.601	0.652
	Base − FE (Folino et al. 2019)	0.567	0.583
	Base − S	0.582	0.629
L_1^c	SSOP − MTL	**0.761**	**0.693**
	SSOP − PT (Folino et al. 2019)	0.658	0.652
	Base − FE (Folino et al. 2019)	0.452	0.438
	Base − S	0.646	0.604
L_1^d	SSOP − MTL	**0.819**	**0.592**
	SSOP − PT (Folino et al. 2019)	0.718	0.515
	Base − FE (Folino et al. 2019)	0.527	0.471
	Base − S	0.734	0.546
L_2	SSOP − MTL	**0.957**	**0.708**
	SSOP − PT (Folino et al. 2019)	0.953	0.666
	Base − FE (Folino et al. 2019)	0.878	0.543
	Base − S	0.955	0.631
L_3	SSOP − MTL	**0.744**	**0.800**
	SSOP − PT (Folino et al. 2019)	0.725	0.799
	Base − FE (Folino et al. 2019)	0.716	0.790
	Base − S	0.710	0.783

The best score for each dataset is shown in bold

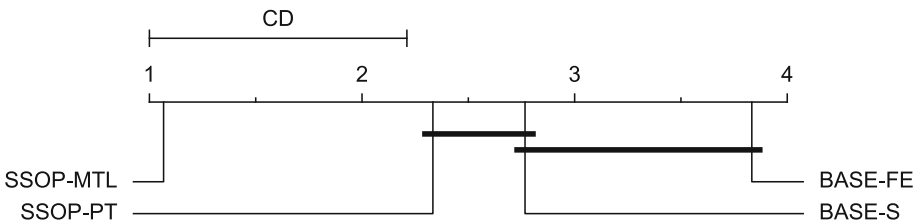


Fig. 4 Critical difference (CD) diagram, drawn on the basis of the results of the Freidman-Nemenyi test, performed on the *AUC* scores obtained by the methods across all the values of *label*% in $\{2.5\%, 5\%, 10\%\}$ and all the datasets. Pairs of methods resulting not significantly different according to the test (i.e., receiving a *p-value* ≥ 0.05) are connected through a horizontal line

label-scarce scenario ...). To this end, we ran all these methods on different data views, obtained by making the percentage *label*% of visible outcome labels range over $\{2.5\%, 5\%, 10\%\}$, for each dataset.

A summarized view of the results obtained is offered by Table 2, which reports the average performance scores achieved (for each dataset) by the methods, and in Fig. 4, showing a *critical difference* (CD) diagram (Demsar 2006; Garcia and Herrera 2009) for the methods. This CD diagram was drawn considering: (i) the average *AUC*-based rankings of the methods (across all the tested combinations of datasets and values of *label*%), and (ii) the result of applying the Friedman-Nemenyi procedure (with a significance level of 0.05) to the series of *AUC* scores achieved by the methods – precisely, the scores obtained by each method, across different datasets and values of *label*%, were considered as the population of results of the method.

Assessing HP′ – the semi-supervised competitors fail to improve the supervised baseline significantly: We start our analysis of test results by comparing the behavior of the two competitors SSOP − PT and Base − FE with that of the supervised baseline Base − S. Looking at scores in Table 2, it seems that the semi-supervised method SSOP − PT proposed in Folino et al. (2019) tends to somewhat improve the baseline (in terms of both *ACC* and *AUC*), but this does not happen over each of the datasets and/or in a very neat way. In particular, the average performances of SSOP − PT are worse than those of Base − S

over the datasets L_1^c and L_2 (if considering the *ACC* metrics only, in the latter case).

The behavior of method Base − FE (implementing a naïve feature-extraction solution) is even more disappointing: not only it scored lower than SSOP − PT over all the datasets, but it performed even worse than the baseline Base − S over all the datasets but L_3.

The diagram in Fig. 4 allows us to draw two observations for these competitors: (i) though SSOP − PT achieved an average rank of 2.33 in this series of tests, and performed significantly better than Base − FE (in line with the conclusions of Folino et al. (2019)), from a statistical viewpoint the performances of SSOP − PT are not significantly different from those of the baseline Base − S (receiving an average rank of 2.77); (ii) despite using some form of pre-training, the naive method Base − FE is not significantly better than the baseline Base − S, and even obtained an average rank (namely, 3.83) worse than that of the baseline.

Clearly, these results provide statistically-significant evidence in support of our hypothesis *HP′*. In particular, the bad behavior of Base − FE allows us to affirm that a pure feature-extraction approach is unsuitable for our problem setting: the representations learnt by using the sole next-activity prediction cannot be reused as a fixed set of features for the traces. In light of this result, we will disregard Base − FE in the rest of our empirical study.

Assessing HP″ – the proposed method SSOP − MTL *outperforms all the other ones significantly:* We now move the focus of our analysis onto the method SSOP − MTL proposed in this work, as a more sophisticated solution than the pre-training schemes explored in Folino et al. (2019). The average scores in Table 2 confirm that SSOP − MTL always outperforms SSOP − PT, Base − FE and Base − S over each of the datasets. Only the difference between the *AUC* scores of SSOP − MTL and SSOP − PT over L_3 somewhat deviates from this pattern. This may be explained by the fact that over this "easy" dataset all the methods tend to perform quite well. However, it is worth noting that even on L_3, SSOP − MTL outperformed SSOP − PT in terms of *ACC*, which is an appropriate metric for making such a fine-grain comparison, since this dataset has a balanced distribution of the outcome classes (differently than the other datasets).

Interestingly, the CD diagram in Fig. 4 clearly shows that SSOP − MTL is by far the best performing method (with an average rank of 1.07), and that its *AUC* performances are significantly different from those of all the other methods.

The findings presented above provide a statistically significant support of the core hypothesis *HP"*, showing that, in scenarios where the fraction of outcome-labeled traces is small, the method SSOP − MTL proposed in this work is neatly superior to both the supervised baseline Base − S and the two existing semi-supervised methods SSOP − PT and Base − FE. In other words, this experimental analysis substantiates our claim that the synergistic multi-task learning strategy implemented in SSOP − MTL manages to counter-balance the negative effect of label scarcity in a more neat and more stable way than simply using next-activity prediction within a pre-training discovery scheme – beside showing that this scheme may even fail to ensure significant improvements over traditional fully-supervised approaches.

6.4.2 *Detailed Results: Performance Scores Obtained with all the Outcome-Label Percentages*

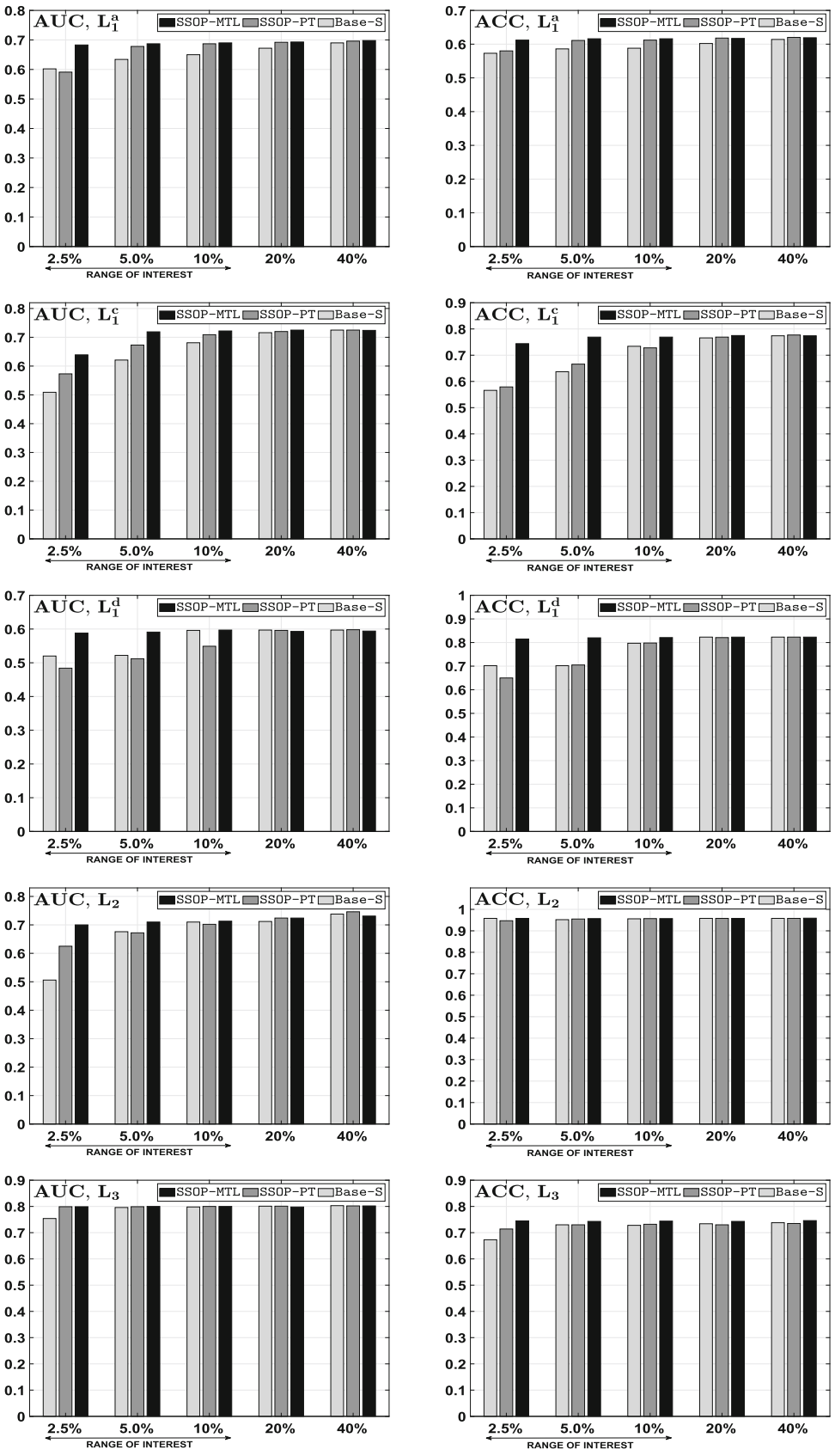
We next analyze the behavior of methods SSOP − MTL, SSOP − PT and Base − S (disregarding the bad-performing method Base − FE, for the sake of brevity) for all the values of *label*% in $\{2.5\%, 5\%, 10\%, 20\%, 40\%\}$. The scores obtained on each dataset, in terms of both *AUC* (left) and *ACC* (right), are shown in Fig. 5 in the form of a bar chart – the better the score achieved by a method, the higher its associated bar.

We are mainly interested in investigating the *AUC* trend of each method when *label*% stays under the critical threshold of 10% delimiting the critical label-scarce range (named "range of interest" in Fig. 5) for it. Higher percentages of *label*% are used here to develop a trend analysis for the methods.

From Fig. 5, it clearly emerges that all the discovery methods tend to perform quite similarly when using a sufficiently large amount of labeled data (i.e., for *label*% $\in \{20\%, 40\%\}$), but the differences among them become evident when entering the critical range *label*% $\leq 10\%$, as expected.

In particular, no matter the evaluation metric and log, our method SSOP − MTL always outperforms both SSOP − PT and Base − S in the range of interest – even though the differences in terms of *AUC* are not very neat on L_3, in line with what observed previously. Anyway, even on L_3, the *AUC* score of SSOP − MTL is substantially higher than that of Base − S in when using the lowest percentage of labeled data (*label*% $= 2.5\%$), and surpasses all the methods in terms of *ACC* no matter the value of *label*% – let us remind that *ACC* is a proper evaluation metric for balanced logs like L_3, while it is rather misleading on imbalanced ones like L_2.

Observing Fig. 5, two facts confirm the compelling robustness of SSOP − MTL to data scarcity: (1) the performance of SSOP − MTL tend to remain steady over a wider range of values of *label*%, compared to the other methods; and (2) even when only 2.5% of the traces are labeled, the


AUC, L_1^a
ACC, L_1^a
AUC, L_1^c
ACC, L_1^c
AUC, L_1^d
ACC, L_1^d
AUC, L_2
ACC, L_2
AUC, L_3
ACC, L_3
SSOP-MTL
SSOP-PT
Base-S
2.5%
5.0%
10%
20%
40%
RANGE OF INTEREST

◀**Fig. 5** *AUC* and *ACC* obtained, for different amounts of labeled data (namely, for $label\% \in \{2.5\%, 5\%, 10\%, 20\%, 40\%\}$), by $\texttt{SSOP} - \texttt{MTL}$, $\texttt{SSOP} - \texttt{PT}$ and $\texttt{Base} - \texttt{S}$ on each of the datasets. Each chart is annotated with the evaluation metric and dataset it refers to, for the sake of readability

performance degradation of $\texttt{SSOP} - \texttt{MTL}$ is neatly lower than those of $\texttt{Base} - \texttt{S}$ and $\texttt{SSOP} - \texttt{PT}$ (if excluding the exceptional case of L_3).

Minor improvements with respect to $\texttt{Base} - \texttt{S}$ are obtained by the competitor semi-supervised method $\texttt{SSOP} - \texttt{PT}$, within our range of interest. Indeed, if $\texttt{SSOP} - \texttt{PT}$ continues achieving pretty better *AUC* scores than the baseline method on the datasets L_1^a, L_1^c and L_2, the opposite happens on dataset L_1^d. Again, on L_3, $\texttt{SSOP} - \texttt{PT}$ works as better as $\texttt{SSOP} - \texttt{MTL}$ in terms of *AUC* (but not in terms of *ACC*), whereas the *ACC* score of $\texttt{SSOP} - \texttt{PT}$ on L_2 goes slightly below that of $\texttt{Base} - \texttt{S}$ when $label\% = 2.5\%$.

Finally, we observe that the superiority of $\texttt{SSOP} - \texttt{MTL}$ over the supervised baseline $\texttt{Base} - \texttt{S}$ is not a trivial result. In fact, semi-supervised learning methods are known to possibly lead to performance degradation when using too many and/or misleading unlabeled data (Van Engelen and Hoos 2020). Interestingly, this does not seem to be the case for $\texttt{SSOP} - \texttt{MTL}$, which always outperforms $\texttt{Base} - \texttt{S}$, even when the fraction of outcome-labeled data is very small. The ability of $\texttt{SSOP} - \texttt{MTL}$ to also outperform the existing semi-supervised approaches to outcome prediction (and, in particular, the pre-training method $\texttt{SSOP} - \texttt{PT}$ proposed in Folino et al. (2019)) makes our proposal even more significant in the current research landscape.

7 Conclusion, Discussion and Future Work

7.1 Summary: Main Contributions and Findings

We have presented a semi-supervised learning method, named $\texttt{SSOP} - \texttt{MTL}$, for discovering a DNN-based outcome predictor in scenarios where a limited number of log traces are equipped with a ground-truth outcome-class label. The method leverages a novel multi-task-learning approach, which employs: (i) a multi-target DNN model addressing the prediction of both outcomes and next-activities jointly; (ii) a multi-objective training algorithm that changes the weights of the two loss functions (one per task) dynamically, in order to balance the opposite goals of fitting the outcome prediction task and of conserving the data-representation skills learnt with the auxiliary task.

Experiments on several real-life logs allowed us to analyze this method, and to obtain the following two core findings: (i) both pre-training methods $\texttt{SSOP} - \texttt{PT}$ and $\texttt{Base} - \texttt{FE}$ defined in Folino et al. (2019) fail to improve the baseline significantly across different datasets; (ii) the proposed method $\texttt{SSOP} - \texttt{MTL}$ significantly outperforms both the baseline and these competitors (i.e., $\texttt{SSOP} - \texttt{PT}$ and $\texttt{Base} - \texttt{FE}$).

7.2 Validity Threats and Limitations

In general, extending a supervised learning system with mechanisms for exploiting unlabeled data is not guaranteed to always improve the system, since there may be application domains where the bias coming from these data can be useless or even misleading, with respect to the target task that the system is ultimately meant to accomplish (Van Engelen and Hoos 2020; Ouali et al. 2020). Moreover, in extreme scenarios where even the number of unlabeled traces available is small, the auxiliary task itself can incur overfitting, and exert negative representation-degeneration effects (Liu et al. 2021). We could have included smaller logs (e.g., the *sepsis* and *production* logs considered in Teinemaa et al. (2019)) in our experimental analysis, in order to provide the reader with examples of such challenging application scenarios. However, for the sake of presentation, we decided to exclude such straightforward use cases, since it does not make much sense to apply complex DNN predictors to small datasets, for which traditional ML methods have been proven to work appropriately (Teinemaa et al. 2019, 2018). On the other hand, there may be domains in which forecasting the outcome class is such an easy task that even very few labeled example traces suffice to train a good predictor. In such a case, where supervised-learning methods work well, a semi-supervised approach hardly brings substantial improvements – this seems to be the case with dataset L_3 in our experiments.

Though the set of logs used in our experiments is limited (in part for the sake of reproducibility and comparison with relevant methods tested in Teinemaa et al. (2018) and in Teinemaa et al. (2019)), we believe that our experimental findings represent a significant novel contribution to current literature. In particular, it was interesting to discover that the pure pre-training method $\texttt{SSOP} - \texttt{PT}$ (Folino et al. 2019) is not significantly better than the fully-supervised baseline $\texttt{Base} - \texttt{S}$ across all the datasets. This result, together with the weak performance observed for the feature-extraction method $\texttt{Base} - \texttt{FE}$, shows that not all semi-supervised strategies are a valuable solution for the problem studied here. The limited effectiveness of $\texttt{SSOP} - \texttt{PT}$ in confronting the lack of labeled examples is likely to descend from the fact that, on some of the datasets, it may have incurred representation-degeneration and/or catastrophic-forgetting phenomena (Liu et al. 2021). Thus, it is safe to regard our experimental study as an appropriate way

of showing that there are several real label-scarce scenarios where a well-devised semi-supervised discovery like SSOP − MTL can find better outcome predictors than methods which train the same kind of DNN architecture by using a traditional supervised approach or a pre-training one.

Even though our approach is parametric with respect to the internal structure of the encoder sub-net, in our experimental study we instantiated it with a stack of LSTM layers, which is a consolidated popular solution in the field. However, in principle, the performances of the methods analyzed in this work might vary when using a different kind of neural architecture for the encoder.

Minor validity threats may descend from the fact that we did not perform an optimal tuning of the hyperparameters in the DNN models, but rather chose them based on the best-working setting identified in Teinemaa et al. (2018). This allowed us to find an optimal configuration for the fully-supervised baseline *Base-S*, which is not ensured to also be the most appropriate choice for the multi-target DNN architecture M^{O+A} employed in the semi-supervised method SSOP − MTL. Thus, if combining method SSOP − MTL with ad-hoc hyperparameter optimization, the method could perform better than in our experimental study.

7.3 Implications and Future Work

As to the practical implications of our work, we are confident that it can allow for widening the application scope of DNN-based outcome predictors, encompassing other relevant contexts (e.g., concerning the prediction of faults, fraud, normative-compliance violations, missed customer-satisfaction objectives) where the outcome-labeled traces available are not sufficient to train a good prediction model in a supervised way. Replacing these methods with our semi-supervised method is a more convenient and cheaper solution for enterprises and organizations, compared to trying to expand the set of outcome-labeled log traces (e.g., by directing more efforts towards manual auditing or towards customers surveys).

We hope that this work will stimulate theoretical and experimental research on the challenging outcome-prediction setting considered here. In particular, we foresee the following promising lines of research: (i) studying the combination of the proposed semi-supervised approach with other kinds of encoders (e.g., based on GRU (Hinkka et al. 2018), convolutional (Pasquadibisceglie et al. 2019) or Transformer (Vaswani et al. 2017) components); (ii) devising and testing alternative approaches to the problem, e.g., employing different auxiliary tasks than next-activity prediction. (iii) investigating which characteristics (e.g., number of labeled data available, degree of complexity/flexibility of the business process) of an outcome-prediction scenario make it more or less suitable for the application of semi-supervised learning solutions.

7.4 Test Reproducibility

The methods and the evaluation procedure employed in the experiments of Sect. 6 were implemented in *Python 3.7.7*, using *Keras* and *TensorFlow 2.0* APIs. Specifically, two distinct code packages were developed to implement the proposed method SSOP − MTL, and the remaining ones, respectively. All the test results in Sect. 6 can be reproduced by running these packages on a local copy of the datasets described in that section. Instructions and material for reproducing the tests can be found at https://github.com/francescofolino/semi-supervised_outcome_prediction.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12599-022-00749-9.

References

Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155

Camargo M, Dumas M, González-Rojas O (2019) Learning accurate LSTM models of business processes. In: Intl. conf. on business process management (BPM). Springer, Heidelberg, pp 286–302

Chan DY, Vasarhelyi MA (2018) Innovation and practice of continuous auditing. In: Continuous auditing. Emerald, Bingley

Dai AM, Le QV (2015) Semi-supervised sequence learning. In: Proc of the 29th intl. conf. on neural inform. processing systems (NIPS), pp 3079–3087

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Evermann J, Rehse JR, Fettke P (2017) Predicting process behaviour using deep learning. Decis Support Syst 100:129–140

Fazzinga B, Folino F, Furfaro F, Pontieri L (2020) An ensemble-based approach to the security-oriented classification of low-level log traces. Expert Syst Appl 153(113):386

Folino F, Folino G, Guarascio M, Pontieri L (2019) Learning effective neural nets for outcome prediction from partially labelled log data. In: 31st IEEE intl. conf. on tools with artificial intelligence (ICTAI 2019), pp 1396–1400

French RM (1999) Catastrophic forgetting in connectionist networks. Trends Cogn Sci 3(4):128–135
Garcia S, Herrera F (2009) An extension on "statistical comparisons of classifiers over multiple data sets'' for all pairwise comparisons. J Mach Learn Res 9:2677–2694
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT, Cambridge
Hashmi M, Governatori G, Lam HP, Wynn MT (2018) Are we done with business process compliance: state of the art and challenges ahead. Knowl Inf Syst 57(1):79–133
Hinkka M, Lehto T, Heljanko K, Jung A (2018) Classifying process instances using recurrent neural networks. In: Conf. on business process management (BPM), pp 313–324
Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
Käppel M, Jablonski S, Schönig S (2021) Evaluating predictive business process monitoring approaches on small event logs. In: Paiva ACR, Cavalli AR, Ventura Martins P, Pérez-Castillo R (eds) Quality of information and communications technology, pp 167–182
Kiros R, Zhu Y, Salakhutdinov R, Zemel RS, Torralba A, Urtasun R, Fidler S (2015) Skip-thought vectors. In: Proc. of the 28th intl. conf. on neural inf. processing systems (NIPS), vol 2, pp 3294–3302
Kratsch W, Manderscheid J, Röglinger M, Seyfried J (2020) Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. Bus Inf Syst Eng 63(3):261–276
Laine S, Aila T (2016) Temporal ensembling for semi-supervised learning. arXiv:1610.02242
Lin B, Jones CA (1997) Some issues in conducting customer satisfaction surveys. J Mark Pract Appl Mark Sci 3(1):4–13
Lin L, Wen L, Wang J (2019) Mm-pred: a deep predictive model for multi-attribute event sequence. In: Proc. of the 2019 siam intl. conf. on data mining. SIAM, pp 118–126
Liu X, Zhang F, Hou Z, Mian L, Wang Z, Zhang J, Tang J (2021) Self-supervised learning: generative or contrastive. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2021.3090866
Maggi FM, Di Francescomarino C, Dumas M, Ghidini C (2014) Predictive monitoring of business processes. In: Proc. of the 26th conf. on adv. inf. syst. eng. (CAiSE), pp 457–472
Mehdiyev N, Evermann J, Fettke P (2018) A novel business process prediction model using a deep learning method. Bus Inf Syst Eng 62(2):1–15
Metzger A et al (2015) Comparing and combining predictive business process monitoring techniques. IEEE Trans Syst Man Cybern Syst 45(2):276–290
Metzger A, Neubauer A, Bohn P, Pohl K (2019) Proactive process adaptation using deep learning ensembles. In: Proc. of the 31st intl. conf. on adv. inf. syst. eng. (caise), pp 547–562
Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv:1301.3781
Miyato T, Si Maeda, Koyama M, Ishii S (2018) Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE Trans Pattern Anal Mach Intell 41(8):1979–1993
Navarin N, Vincenzi B, Polato M, Sperduti A (2017) LSTM networks for data-aware remaining time prediction of business process instances. In: IEEE symp. series on comp. intelligence (SSCI), pp 1–7
Nolle T, Luettgen S, Seeliger A, Mühlhäuser M (2018) Analyzing business process anomalies using autoencoders. Mach Learn 107(11):1875–1893
Ouali Y, Hudelot C, Tami M (2020) An overview of deep semi-supervised learning. arXiv:2006.05278
Pasquadibisceglie V, Appice A, Castellano G, Malerba D (2019) Using convolutional neural networks for predictive process analytics. In: Intl. conf. on process mining (icpm), pp 129–136
Qiu X, Sun T, Xu Y, Shao Y, Dai N, Huang X (2020) Pre-trained models for natural language processing: a survey. Sci China Technol Sci 63(10):1872–1897
Rasmus A, Berglund M, Honkala M, Valpola H, Raiko T (2015) Semi-supervised learning with ladder networks. In: Proc. of the 28th intl. conf. on neural inf. processing systems (NIPS), pp 3546–3554
Seeliger A, Luettgen S, Nolle T, Mühlhäuser M (2021) Learning of process representations using recurrent neural networks. In: Proc. of 33rd intl. conf. on adv. inf. systems eng. (caise). Springer, pp 109–124
Sheridan RP (2013) Time-split cross-validation as a method for estimating the goodness of prospective prediction. J Chem Inf Model 53(4):783–790
Tarvainen A, Valpola H (2017) Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results. In: Proc. of the 31st intl. conf. on neural inf. processing systems (NIPS), pp 1195–1204
Tax N, Verenich I, La Rosa M, Dumas M (2017) Predictive business process monitoring with LSTM neural networks. In: Proc. of the 29th intl. conf. on adv. inf. syst. eng. (CAiSE), pp 477–492
Taymouri F, La Rosa M, Erfani S, Bozorgi ZD, Verenich I (2020) Predictive business process monitoring via generative adversarial nets: the case of next event prediction. In: Proc. of intl. conf. on business process management (BPM), pp 237–256
Teinemaa I, Dumas M, Leontjeva A, Maggi FM (2018) Temporal stability in predictive process monitoring. Data Min Knowl Discov 32(5):1306–1338
Teinemaa I, Dumas M, Rosa ML, Maggi FM (2019) Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans Knowl Discov Data 13(2):17:1–17:57
Triguero I, García S, Herrera F (2013) Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. Knowl Inf Syst 42:245–284
Van Der Aalst W (2011) Process mining: discovery, conformance and enhancement of business processes. Springer, Heidelberg
Van Engelen JE, Hoos HH (2020) A survey on semi-supervised learning. Mach Learn 109(2):373–440
Vaswani A et al (2017) Attention is all you need. In: Proc of the 31st intl. conf. on neural inform. processing systems (NIPS), pp 5998–6008
Xu K, Zhang M, Li J, Du SS, Kawarabayashi K, Jegelka S (2020) How neural networks extrapolate: from feedforward to graph neural networks. arXiv:2009.11848