JARVIS: Joining Adversarial Training with Vision Transformers in Next-Activity Prediction

Vincenzo Pasquadibisceglie, and Annalisa Appice, and Giovanna Castellano, *Senior Member, IEEE* and Donato Malerba, *Senior Member, IEEE*

Abstract—In this paper, we propose a novel predictive process monitoring approach, named JARVIS, that is designed to achieve a balance between accuracy and explainability in the task of next-activity prediction. To this aim, JARVIS represents different process executions (traces) as patches of an image and uses this patch-based representation within a multi-view learning scheme combined with Vision Transformers (ViTs). Using multi-view learning we guarantee good accuracy by leveraging the variety of information recorded in event logs as different patches of an image. The use of ViTs enables the integration of explainable elements directly into the framework of a predictive process model trained to forecast the next trace activity from the completed events in a running trace by utilizing self-attention modules that give paired attention values between two picture patches. Attention modules disclose explainable information concerning views of the business process and events of the trace that influenced the prediction. In addition, we explore the effect of ViT adversarial training to mitigate overfitting and improve the accuracy and robustness of predictive process monitoring. Experiments with various benchmark event logs prove the accuracy of JARVIS compared to several current state-of-the-art methods and draw insights from explanations recovered through the attention modules.

Index Terms—Predictive process monitoring, Deep learning, Multi-view learning, Adversarial training, Vision transformers, Attention, XAI, Computer vision

1 Introduction

Predictive Process Monitoring (PPM) techniques allow the extraction of smart knowledge from historical, raw event data of business processes, in order to enable the prediction of future states (e.g., next-activity, completion time, outcome) of a process execution given its running trace and raw event data of historical traces as inputs.

Over the past five years, the predominance of deep learning in predictive modeling has been increasingly assessed in PPM systems. Several deep neural networks, e.g., LSTMs [1], [2], [3], CNNs [4], [5], GANs [6] and Autoencoders [7], have recently contributed to gaining accuracy into PPM systems thanks to their capability to learn accurate deep neural models that can enable proactive and corrective actions to improve process performance and mitigate risks. However, deep neural networks generally learn opaque models implicitly represented in form of a huge number of numerical weights that are difficult to explain due to the complexity of the network structure. The opacity of deep neural models is acceptable as long as accuracy was the dominant criterion for assessing the quality of PPM systems. Despite the priority of PPM systems today is still to provide accurate predictions of future states of running traces, easier-to-explain predictive models are becoming increasingly desirable in PPM applications.

The explainability of predictive models refers partially to how easily humans may comprehend their underlying assumptions and reasoning. Explanations may disclose measurable factors on which trace characteristics influence the prediction of a future state of a running trace and to what extent. Explaining the effect of certain trace characteristics on trace predictions can contribute to translating predictions into some domain knowledge and allow PPM systems to better benefit from the trust of process stakeholders. Recent studies [8], [9], [10] have investigated the use of existing XAI (eXplainable AI) methods to explain opaque PPM models. However, model explainability in the context of deep learning-based PPM systems is yet underexplored.

To bridge the gap between accuracy and explainability in PPM models, in this paper, we propose a novel method for predicting the next-activity of a running trace. The proposed method, called JARVIS (Joining Adversarial tRaining with VISion transformers in next-activity prediction), combines adversarial training with Vision Trasformers (ViT) [11] to achieve a valid balance between the accuracy and the explainability of the model. Explainability is achieved thanks to the adoption of a ViT, i.e., a deep neural architecture composed of multiple self-attention layers that implement an attention mechanism to differentiate the significance of each part of the input sequence of data, thus providing a form of explanation of the model behavior in terms of most informative inputs. Accuracy improvement is faced via adversarial training [12] which incorporates perturbed (i.e., adversarial) inputs into the training process, in order to mitigate overfitting and improve generalization [13]. Moreover, boosted by our recent results [3], we also adopt multi-view learning [14], which consists in learning with multiple sources (views) of event information (e.g., activity, resource, timestamp, cost) to improve the generalization performance. Specifically, in JARVIS events are seen as multiple views of

• *V. Pasquadibisceglie, A. Appice, G. Castellano and D. Malerba are with the Dipartimento di Informatica, Università degli Studi Aldo Moro di Bari and Consorzio Interuniversitario Nazionale per l'Informatica - CINI, via Orabona, 4 - 70125 Bari - Italy E-mail: vincenzo.pasquadibisceglie@uniba.it, annalisa.appice@uniba.it, giovanna.castellano@uniba.it, donato.malerba@uniba.it*

the same running trace, and this enables handling multiple characteristics of a trace.

Another key feature of JARVIS is a novel representation of running traces as images. Specifically, each view of a trace is encoded as a separate image patch (i.e., a square-shaped region within an image) that encodes the sequence of embedded events in the view. The embedding representation of events in each view is derived by accounting for neighbour information of each event in the running trace (i.e., view information of events that precede the considered event in the running trace). Notably, this image representation differs from the ones adopted in [4], [5] which lose information about the order of trace events due to an aggregation step. In addition, the novel representation proposed in JARVIS allows the embedding of raw event data belonging to any view. It also differs from the imagery representation introduced in [15] that represents full traces populated with partially labeled events and is used to predict the completion time of a target event within a full trace by accounting for attributes of all events in the trace and completion times of events preceding the target one. Conversely, the imagery representation adopted in JARVIS is used to predict the next-activity of the running trace without having any information on the next event.

In short, the contributions of this paper are as follows.

- The formulation of a novel multi-view learning approach that performs an image-like engineering step to represent running traces as image patches and trains a ViT model for next-activity prediction, in order to learn relationships between multiple input views and provide explainability in form of attention of the model on significant inputs.
- The use of attention maps of ViT models as a form of explanation to disclose the effect that specific views and intra-view events (i.e., characteristics of trace events in a specific view) may have on the PPM model's reasoning.
- The exploration of adversarial training as a learning strategy to mitigate overfitting and improve ViT model generalization.
- The presentation of the results of an in-depth evaluation examining the ability of our approach to achieve accuracy comparable to competitive deep learning-based approaches drawn from the recent literature on PPM systems, as well as the accuracy of the proposed learning components.

The paper is organized as follows. Section 2 overviews recent advances of PPM literature in the next-activity prediction of business processes and XAI. Preliminary concepts are reported in Section 3, while the proposed JARVIS approach is described in Section 4. The experimental setup is illustrated in Section 5. The results of the evaluation of the proposed approach are discussed in Sections 6 and 7, regarding accuracy and explainability, respectively. Finally, Section 8 recalls the purpose of our research, draws conclusions, and illustrates possible future developments.

2 Related Work

The JARVIS approach proposed in this paper is a deep learning-based method that resorts to a trace engineering step to represent running traces as image patches and neural attention mechanisms for yielding next-activity predictions of running traces equipped with explainability. Therefore, the literature overview is organized on two fronts: on one side, we focus on recent PPM studies on next-activity prediction using deep learning (Section 2.1) and on the other side we overview preliminary attempts to use XAI techniques in the realm of PPM (Section 2.2). A summary of the characteristics of the main related methods discussed in the following SubSections is reported in Table 1.

2.1 Next-activity prediction

Several recent PPM studies have investigated the performance of various deep neural network architectures (e.g., LSTMs, RNNs, CNNS) for predicting the next-activity of a running trace. These studies commonly adopt an encoding step that is the responsible for mapping complex, categorical, event data information into a numerical feature space, in order to feed the deep neural network. These studies mainly adopt data engineering methods like One-Hot-Encoding (OHE) (e.g., [1]) or embeddings (e.g., [2], [16]) to map running traces into real-valued vectors. Notably the use of the embedding representation of a categorical input allows us to avoid a sparse representation of categorical input data that is one of the limits of the OHE representation. [24] compares several embedding mechanisms, i.e., Act2Vec, Trace2Vec, Log2Vec, and Model2Vec, which perform data engineering at the level of activities, traces, logs, and models. [25] uses Word2Vec to learn a context-aware real-valued representation of categorical input (e.g., activities, resources), which can be easily fine-tuned to the new events recorded in a streaming scenario. [26] evaluates the performance of several graph embeddings. [1] augments the OHE of activities with several numerical features extracted from the timestamp information such as the time of day, the time since the previous event, the time since the beginning of the trace, while [2] uses embedding networks and handcrafted features. Image-like data engineering methods have been introduced by [4], [20]. In particular, [20] proposes an approach to transform prefix traces into grey-scale images by mapping the activity and timestamp information of each prefix trace into two grey-scale pixel columns. [4] describes an imagery encoding approach that first extracts a numeric feature vector representation of prefix traces by aggregating activity, resource and timestamp information. Then it transforms the feature vector representation of a prefix trace into a color-scale image by mapping every numeric feature into a RGB pixel. Notably, [4] uses a color imagery encoding of running traces as in our work. However, it looses information about the order of trace events due to the aggregation step. In addition, it is defined to encode activity, resource and timestamp information. Differently JARVIS keeps the information on the temporal order of events and it is defined to encode event information from any view. A recent survey of the main progress in trace encoding techniques has been described in [27].

Focusing the attention on the deep neural model, as most of PPM studies consider prefix traces represented as sequences, they adopt sequence-based deep neural network architectures such as LSTMs or RNNs. In particular, [1]

Table 1: A summary of characteristics of related methods

Reference	Views	ML/DL	XAI	Task
[1]	Activity, Time	LSTM	-	Next-activity, Next timestamp, Completion time, Activity suffix
[16]	Activity, Resource	LSTM	-	Next-activity
[2]	Activity, Resource, Timestamp	LSTM	-	Next-activity, Next timestamp, Next role, Completion time, Activity suffix
[3]	All	LSTM	-	Next-activity
[17], [18], [19]	All	Decision Tree, Naive Bayes, DNN	-	Deviant traces
[20]	Activity, Timestamp	CNN	-	Next-activity
[4]	Activity, Resource, Timestamp	CNN	-	Next-activity
[8]	Activity, Resource, Role	LSTM	SHAP	Completion time, Activity occurrence, User satisfaction
[10]	Activity, Resource, Role	CatBoost, LSTM	SHAP	Completion time, Activity occurrence, User satisfaction
[21]	Activity, Resource, Timestamp	ANFIS	Neuro-fuzzy rules	Trace ouctcome
[22]	Activity	Gated Graph Neural Network	Graphs	Trace ouctcome
[23]	Activity	Transformer	Attention maps	Next-activity
[9]	Activity, Resource, Timestamp	Attention+LSTM	Attention maps	Next-activity

describes an LSTM-based PPM method to predict both the next-activity and the completion time of a prefix trace. It uses OHE of activity and timestamp information. [16] illustrates a LSTM-based PPM method that accounts for embeddings of both activities and resources, but neglects timestamps. Also [2] describes a PPM method that trains an LSTM architecture to predict sequences of next events, their timestamps and their associated resource pool. This method accounts for activity and resource by resorting to a pre-trained embedding representation of the categorical information. [3] proposes a multi-input LSTM architecture that is able to process all possible views present in an event log. This flexible multi-view method has the advantage of capturing the possible information among the views and increase the predictive model's accuracy performance. The embeddings of the categorical information are computed within the multi-input network. Multi-view learning methods are also developed also for deviant trace detection. [17], [18], [19] describe multi-view ensemble-based methods that combine different single-view classifiers to disentangle deviant process instances form normal ones. Specifically, [17] describes the use of decision trees and association rule classifiers as single-view classifiers of the ensemble. [18] explores the combination of single-view Bayesian classifiers, while [19] combines single-view deep neural models.

In spite of the proliferation of LSTM methods for PPM, a few recent studies have started the investigation of computer vision-based approaches in PPM. For example, [20] describes a CNN architecture to predict the next activity of a grey-scale image of a prefix trace, while [4] describes a CNN with Inception to predict the next activity of a RGB image of a prefix trace. Both approaches handle information of apriori defined views under a feature extraction step.

Despite the methods described above achieve good accuracy performance in various problems, they are not equipped with explainability techniques.

2.2 Explainability in PPM

A few recent studies have started the investigation of explainability in deep learning-based PPM models. However, most of the existing work on explainable PPM approaches use post-hoc methods to explain model predictions. For example, [8] describes a PPM framework that uses "post-hoc" explanations generated by the SHAP technique applied to trained black-box LSTM models. Also [10] reports on the use of post-hoc, SHAP explanations for both the LSTM and the CatBoost method. [28] compares and evaluates explanations of process predictions yielded by different post-hoc frameworks (e.g., LIME and SHAP). [29] leverages post-hoc explainers to understand why a PPM model provides wrong predictions, eventually improving its accuracy.

The above post-hoc explainers are model-agnostic, i.e., they can be used independently of the model and they do not have any effect on the model training. On the other hand, a few recent PPM approaches have started focusing on intrinsic explainable deep neural models that directly produce interpretable models. One of the few works in this direction is [21] that presents a fully interpretable PPM model for outcome prediction based on a set of fuzzy rules acquired from event data by training a neuro-fuzzy network. [22] uses gated graph neural networks to visualize how much the different activities included in a process impact the PPM model predictions. [23] incorporates explainability into the structure of a PPM model by replacing the cells with self-attention mechanisms in recurrent neural networks. Similarly [9] develops attention-based models to incorporate explainability directly into the PPM model. Notably, both [23] and [9] use the attention mechanism as in our work. Our approach differs from [23] and [9] since it is able to handle event information of any view without forcing any apriori-defined views at the input level. In addition, JARVIS uses a ViT architecture that incorporates self-attention modules to detect inter-view relationships, in addition to intra-view event information. We use the intrinsic attention on inter-view and intra-view information to obtain both local and global explanations of predictions.

3 Preliminary Concepts

Given a business process, a trace describes the life cycle of a process execution in terms of a sequence of events. According to this definition of a trace, an event refers to a complex entity composed of two mandatory characteristics, i.e., the activity and the timestamp (indicating the date and time of the activity occurrence), as well as several optional characteristics, such as the resource triggering the activity or the cost of completing the activity. Based on this definition, an event can be characterized by different views, being a view the description of the event along a specific characteristic. Hence, every event has two mandatory views that are associated with the activity and the timestamp, as well as m

additional views associated with optional characteristics of events. Let $\mathcal{A}$ be the set of all activity names, $\mathcal{S}$ be the set of all trace identifiers, and $\mathcal{T}$ be the set of all timestamps and $\mathcal{V}_j$ with $1 \leq j \leq m$ be the set of all names in the j-th view.

Definition 1 (Event). Given the event universe $\mathcal{E} = \mathcal{S} \times \mathcal{A} \times \mathcal{T} \times \mathcal{V}_1 \times \ldots \times \mathcal{V}_m$, an event $e \in \mathcal{E}$ is a tuple $e = (\sigma, a, t, v_1, \ldots, v_m)$ that represents the occurrence of activity a in trace σ at timestamp t with characteristics $v_1, v_2, \ldots, v_m$.

Let us introduce the functions: $\pi_{\mathcal{S}} \colon \mathcal{E} \mapsto \mathcal{S}$ such that $\pi_{\mathcal{S}}(e) = \sigma$, $\pi_{\mathcal{A}} \colon \mathcal{E} \mapsto \mathcal{A}$ such that $\pi_{\mathcal{A}}(e) = a$, $\pi_{\mathcal{T}} \colon \mathcal{E} \mapsto \mathcal{T}$ such that $\pi_{\mathcal{T}}(e) = t$ and $\pi_{\mathcal{V}_j} : \mathcal{E} \mapsto \mathcal{V}_j$ such that $\pi_{\mathcal{V}_j}(e) = v_j$ and $j = 1, \ldots, m$.

Definition 2 (Trace). Let $\mathcal{E}^*$ denote the set of all possible sequences on $\mathcal{E}$. A trace σ is a sequence $\sigma = \langle e_1, e_2 \ldots, e_n \rangle \in \mathcal{E}^*$ so that: (1) $\forall i = 1, \ldots, n$, $\exists e_i \in \mathcal{E}$ such that $\sigma(i) = e_i$ and $\pi_{\mathcal{S}}(e_i) = \sigma$, and (2) $\forall i = 1, \ldots, n-1, \pi_{\mathcal{T}}(e_i) \leq \pi_{\mathcal{T}}(e_{i+1})$.

Definition 3 (Event log). Let $\mathcal{B}(\mathcal{E}^*)$ denote the set of all multisets over $\mathcal{E}$. An event log $\mathcal{L} \subseteq \mathcal{B}(\mathcal{E}^*)$ is a multiset of traces.

Definition 4 (Prefix trace). A *prefix trace* $\sigma^k = \langle e_1, e_2, \ldots, e_k \rangle$ is the sub-sequence of a trace σ starting from the beginning of the trace σ with $1 \leq k = |\sigma^k| < |\sigma|$.

A trace is a complete (i.e., started and ended) process instance, while a prefix trace is a process instance in execution (also called running trace). The activity $\pi_{\mathcal{A}}(e_{k+1}) = a_{k+1}$ corresponds to the next-activity of σ^k, i.e., $next(\sigma^k) = \pi_{\mathcal{A}}(e_{k+1})$ with $e_{k+1} = \sigma(k+1)$.

Definition 5 (Multiset of labeled prefix traces). Let $\mathcal{L} \subseteq \mathcal{B}(\mathcal{E}^*)$ be an event log, $\mathcal{P} \subseteq \mathcal{B}(\mathcal{E}^* \times \mathcal{A})$ is the multiset of all prefix traces extracted from traces recorded in $\mathcal{L}$. Each prefix trace is labeled with the next-activity associated to each prefix sequence in the corresponding trace so that $\mathcal{P} = [\sigma^k, \pi_{\mathcal{A}}(e_{k+1}) | \sigma \in \mathcal{L} \wedge 1 \leq k < |\sigma|]$.

Definition 6 (Single-view representation of a labeled prefix trace multiset). Let $\mathcal{V}$ be a view (either mandatory, i.e., $\mathcal{V} = \mathcal{A}$ or $\mathcal{V} = \mathcal{T}$, or optional, i.e. $\mathcal{V} = \mathcal{V}_j$ with $j = 1, \ldots, m$), $\Pi \colon \mathcal{E}^* \mapsto \mathcal{V}^*$ be a function such that $\Pi(\sigma^k) = \Pi(\langle e_1, e_2, \ldots, e_k \rangle) = \langle \pi_{\mathcal{V}}(e_1), \pi_{\mathcal{V}}(e_2) \ldots, \pi_{\mathcal{V}}(e_k) \rangle$. $\mathcal{P}_{\mathcal{V}}$ denotes the multiset of the labeled prefix traces of $\mathcal{P}$ as they are represented in the view $\mathcal{V}$, that is, $\mathcal{P}_{\mathcal{V}} = \{\Pi_{\mathcal{V}}(\sigma^k), a_{k+1} | (\sigma^k, \pi(e_{k+1}))) \in \mathcal{P}\}$.

The *next-activity prediction* is a PPM task often addressed as a multi-class classification problem by resorting to machine learning techniques. Let $F \colon \mathcal{E}^* \times \mathbb{R}^{\mu} \mapsto \mathcal{A}$ be a next-activity prediction model with μ real-valued parameters.

Definition 7 (Next-activity prediction hypothesis function). A next-activity hypothesis $H_{F,\Theta}$ of the model F is a function: $H_{F,\Theta} \colon \mathcal{E}^* \mapsto \mathcal{A}$ with $\Theta \in \Re^{\mu}$ such that $H_{F,\Theta}(x) \approx F(x, \Theta)$.

Definition 8 (Next-activity prediction). Let us consider $H_{F,\Theta} \colon \mathcal{A}^* \mapsto \mathcal{A}$, and σ^k a prefix trace. $H_{F,\Theta}(\sigma^k)$ predicts the expected next-activity of σ^k.

The hypothesis $H_{F,\Theta}$ can be learned from a labeled prefix trace multiset $\mathcal{P}$ by an algorithm that estimates Θ minimizing a cost function $C_{H_{F,\Theta}} \colon \mathcal{E}^* \times A \mapsto \mathbb{R}$, where $C_{H_{F,\Theta}}(\sigma^k, a_{k+1})$ measures the penalty of an incorrect prediction done through $H_{F,\Theta}(\sigma^k)$ for the next-activity a_{k+1}. $H_{F,\Theta}$ depends on the model type. In this study, we represent the labeled multiset as a collection of color image patches that are given as input to a ViT architecture [11].

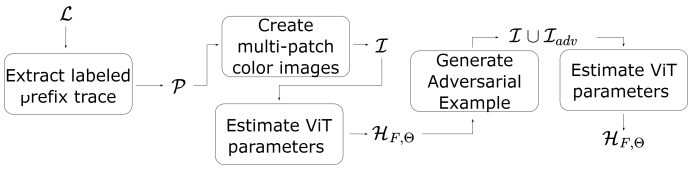

Figure 1: JARVIS pipeline

4 Proposed Approach

In this section we introduce JARVIS, a novel PPM approach for next-activity prediction that combines multi-view learning and adversarial training with Vision Trasformers (ViT) to achieve a valid balance between accuracy and explainability. The proposed approach is schematized in Figure 1. Initially, the labeled prefix trace multiset $\mathcal{P}$ is extracted from the event log $\mathcal{L}$ and transformed into a set $\mathcal{I}$ of multi-patch color images. Then, $\mathcal{I}$ is fed into a ViT architecture that is trained with adversarial training to estimate parameters of a next-activity prediction hypothesis function $H_{F,\Theta}$.

In the following, we detail the main phases of the proposed approach, namely the multi-patch image encoding (Section 4.1), the ViT adversarial training (Section 4.2), and the extraction of the attention maps (Section 4.3).

4.1 Multi-patch image encoding

This phase takes the event log $\mathcal{L}$ as input and creates the multiset of multi-patch color images $\mathcal{I}$ as output. This phase is composed of four steps: 1) Discretizing numerical view information. 2) Generating labeled prefix traces. 3) Training an embedding for prefix traces represented in each view. 4) Transforming embeddings of prefix traces represented in each view into multi-patch color images according to the embeddings trained on the multiple views.

According to the multi-view formulation introduced in Section 3, every event recorded in $\mathcal{L}$ is a complex entity whose representation takes into account two mandatory characteristics (activity $\mathcal{A}$ and timestamp $\mathcal{T}$) and m optional characteristics ($\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m$), respectively. Mandatory and optional views may describe both categorical characteristics (e.g., activity or resource names) and numerical characteristics (e.g., timestamp or cost values). In particular, the timestamp information associated with an event is transformed in the time in seconds passed from the beginning of the trace. In this study, every numerical characteristic is converted into a categorical, format by resorting to the equal-frequency discretization algorithm. This algorithm divides all the values of a numeric characteristic recorded in the training set of the event log into bins that cover the same number of observations. The number of discretization bins of a numeric characteristic is set equal to the average number of distinct categories in the original

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3331020

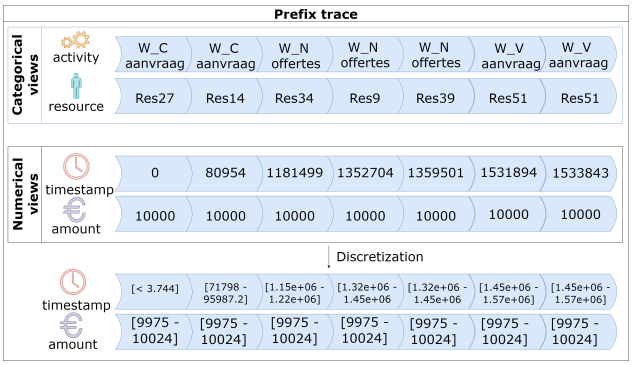

Figure 2: A prefix trace recorded in the event log BPI12WC. It is represented in two categorical views ("activity" and "resource") and two numerical views ("timestamp" and "amount"). Numerical data are transformed into categorical data using the equal-frequency discretization algorithm.

categorical views of the event log. After this step, the event log $\mathcal{L}$ contains all multi-view information in the categorical format. We denote $\mathbf{V}$ the final set of $m+2$ categorical views that characterize events recorded in the pre-processed event log $\mathcal{L}$. For example, Figure 2 shows a sample prefix trace recorded in the event log BPI12WC considered in the experimental study. Event characteristics enclosed in the numeric views "timestamp" and "amount" are transformed into the categorical format through the discretization step. In this way, we obtain a multi-view representation of the prefix trace with all characteristics in the categorical format.

Subsequently, the multiset $\mathcal{P}$ is created by extracting traces from $\mathcal{L}$ and labeling them with the next activity. Since different prefix traces may have different lengths, we use the padding technique in combination with the windowing mechanism to standardize different prefix lengths and populate $\mathcal{P}$ with fixed-sized prefix traces. Let AVG_σ be the average length of all the traces in $\mathcal{L}$, the padding is used with a window length equal to AVG_σ, as in [3]. Prefix traces with length less than AVG_σ are standardized by adding dummy events. Prefix traces with length greater than AVG_σ are standardized by retaining only the most recent AVG_σ events. After this step, $\mathcal{P}$ comprises labeled prefix traces having fixed size equal to AVG_σ.

The Continuous-Bag-of-Words (CBOW) architecture of the Word2Vec scheme [30] is then used to transform the categorical representation of a prefix trace into a bidimensional, numeric embedding representation. The CBOW architecture leverages a feed-forward neural network to predict a target category from the neighbored context. For each view $\mathcal{V} \in \mathbf{V}$, a CBOW neural network, denoted by $CBOW_\mathcal{V}$ is trained in order to convert each single-view sequence $\Pi_\mathcal{V}(\sigma^k) \in \mathcal{P}_\mathcal{V}$ into an AVG_σ-sized numerical vector. Specifically, $\Pi_\mathcal{V}(\sigma^k)$ is converted into a bidimensional, numeric embedding $\mathbf{P}_\mathcal{V} \in \mathbb{R}^{AVG_\sigma \times AVG_\sigma}$ with size $AVG_\sigma \times AVG_\sigma$. For example, Figure 3 shows how the sequence of activities of the sample prefix trace reported in Figure 2 is converted into a 7×7 bidimensional, numeric embedding through Word2Vec. Notice that $AVG_\sigma = 7$ in BPI12WC.

Finally, for each labeled prefix trace $(\sigma^k, a_{k+1}) \in \mathcal{P}$, the list of its multi-view, bidimensional, numeric

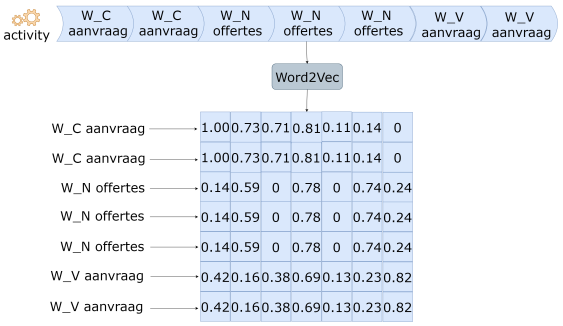

Figure 3: Word2Vec embedding of the sequence of activities enclosed in the sample prefix trace shown in Figure 2

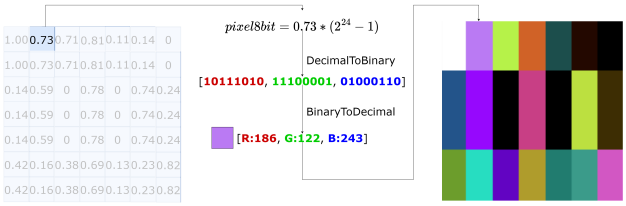

Figure 4: Conversion of the Word2Vec embedding representation shown in Figure 3 into an imagery color patch

embeddings $\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{T}, \ldots, \mathbf{P}_{\mathcal{V}_1}, \ldots, \mathbf{P}_{\mathcal{V}_m}$, generated for $\Pi_\mathcal{A}(\sigma^k), \Pi_\mathcal{T}(\sigma^k), \Pi_{\mathcal{V}_1}(\sigma^k), \ldots, \Pi_{\mathcal{V}_m}(\sigma^k)$, respectively, are converted into the imagery color patches $\mathbf{P}^{\mathbf{rgb}}_\mathcal{A}, \mathbf{P}^{\mathbf{rgb}}_\mathcal{T}, \ldots, \mathbf{P}^{\mathbf{rgb}}_{\mathcal{V}_1}, \ldots, \mathbf{P}^{\mathbf{rgb}}_{\mathcal{V}_m}$ by mapping numeric values of bidimensional embeddings into RGB pixels. In particular, every imagery color patch $\mathbf{P}^{\mathbf{rgb}} \in \mathbb{R}^{AVG_\sigma \times AVG_\sigma \times 3}$ records the embedding of a prefix trace with respect to a view into a numerical tensor with size $AVG_\sigma \times AVG_\sigma \times 3$. Let $\mathbf{P}$ be a bidimensional, numeric embedding, each numeric value of $v \in \mathbf{P}$ is converted into a RGB pixel $v^{\mathbf{rgb}} \in \mathbf{P}^{\mathbf{rgb}}$ by resorting to the RGB-like encoding function adopted in [4]. First, v is scaled in the range $[0, 2^{24} - 1]$. Then the scaled value is considered as the value of a single 24-bit pixel $v^{\mathbf{rgb}}$ with the first eighth bits mapped into the band $\mathbf{R}$, the intermediate eighth bits mapped into the band $\mathbf{G}$ and the last eighth bits mapped into the band $\mathbf{B}$. For example, Figure 4 shows how the Word2Vec embedding of the sequence activities of the prefix trace shown in Figure 3 is converted into an imagery color patch associated with the view "activity".

The $m+2$ color patches of a prefix trace are distributed into a patch grid with size $\lceil\sqrt{m+2}\rceil \times \lceil\sqrt{m+2}\rceil$ from left to right, and from top to bottom. Notice that every cell of the patch grid records a patch with size $AVG_\sigma \times AVG_\sigma \times 3$. In this way, we are able to produce the color image a prefix trace, that is a tensor with size $(\lceil\sqrt{m+2}\rceil \cdot AVG_\sigma) \times (\lceil\sqrt{m+2}\rceil \cdot AVG_\sigma) \times 3$. For example, Figure 5 shows how the four patches, produced for the views "activity","resource", "timestamp" and "amount" of the prefix trace reported in Figure 2, are distributed into a patch grid with size 2×2 forming the final color image of the sample prefix trace. This image has size $14 \times 14 \times 3$ and contains 4 color patches with sizes $7 \times 7 \times 3$ associated with the four views.

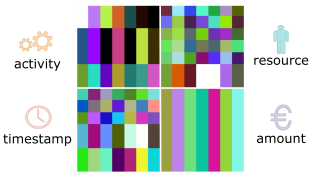

Figure 5: Multi-patch color image with size $14 \times 14 \times 3$. This image is obtained by distributing the four image patches of size $7 \times 7 \times 3$, produced for the views "activity","resource", "timestamp" and "amount" of the prefix trace shown in Figure 2, into a 2×2 patch grid.

The generated multi-patch images are labeled as the corresponding prefix traces and added to the labeled image multiset $\mathcal{I}$.

4.2 ViT architecture

A ViT (Visual Transformer) architecture (fig. 6) is adopted to learn a next-activity prediction hypothesis function $H_{F,\Theta}$ from $\mathcal{I}$ by accounting for intra-patch information and handling inter-patch relationships. We adopt the standard ViT architecture [11] that represents the de-facto standard for computer vision applications and was developed as an extension of the original Transformer architecture originally designed for natural language processing tasks [31]. As the Transformer takes one-dimensional sequences of word embeddings as input, the following four steps are performed to transform the patches into a suitable input for the Transformer:

1) Every multi-patch image $\mathbf{I} \in \mathcal{I}$ is reshaped into the sequence of its $m+2$ flattened 2-dimensional patches $\phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{A}}), \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{T}}), \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{V}_1}), \ldots, \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{V}_m})$ using the the flattening operator $\phi\colon \mathbb{R}^{AVG_\sigma \times AVG_\sigma \times 3} \mapsto \mathbb{R}^{AVG^2_\sigma \cdot 3}$.
2) Every flattened patch sequence is mapped to a constant latent vector size D using a trainable linear projection $\mathbf{E} \in \mathbb{R}^{(AVG^2_\sigma \cdot 3) \times D}$. In this way a sequence of embedded patches, called patch embeddings, is generated.
3) A learnable class embedding, denoted as **class**, is anteposed to the patch embeddings. The value of **class** represents the classification output y.
4) Patch embeddings are augmented with one-dimensional positional embeddings $\mathbf{E}_{pos} \in \mathbb{R}^{(m+3) \times D}$. The positional embeddings enable injection of positional information into the input.

The output of the transformation steps described above is the sequence of embedding vectors defined as follows:

$$\begin{aligned} \mathbf{z_0} \quad = \quad & [\mathbf{class}; \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{A}})\mathbf{E}; \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{T}})\mathbf{E}; \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{V}_1})\mathbf{E}; \ldots; \\ & \phi(\mathbf{P}^{\mathbf{rgb}}_{\mathcal{V}_m})\mathbf{E};] + \mathbf{E}_{pos}. \end{aligned} \tag{1}$$

$\mathbf{z_0}$ feeds the Transformer encoder defined in [31]. This consists of a stack of L identical blocks alternating multi-headed self-attention and MLP blocks, in order to compute:

$$\mathbf{z'_i} \quad = \quad msa(LN(\mathbf{z_{i-1}})) + \mathbf{z_{i-1}}, \tag{2}$$

$$\mathbf{z_i} \quad = \quad mlp(LN(\mathbf{z'_i})) + \mathbf{z'_i}. \tag{3}$$

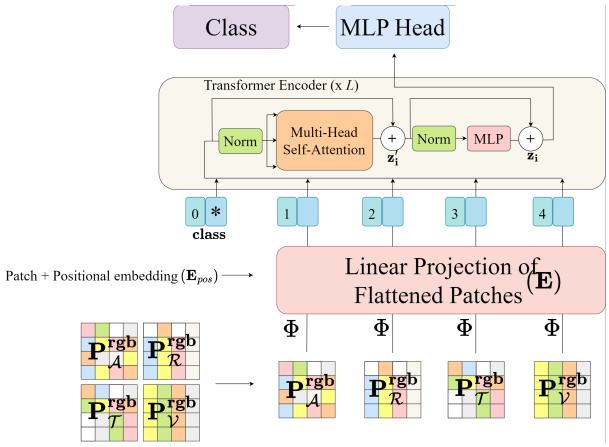

Figure 6: ViT architecture

with $i = 1, \ldots, L-1$, $msa(\bullet)$ the multi-headed self-attention block, $mlp(\bullet)$ the MLP block, $LN(\bullet)$ the Layernorm that is applied before every block. Finally, the value **class** of the L-th block of the encoder output feeds into a MLP classification head with one hidden layer and GELU non-linearity.

The parameters of the ViT architecture are estimated through the adversarial training strategy. Adversarial training is a well known learning strategy in which the training set is augmented with adversarial samples generated from the training set by perturbing some samples. Adversarial training is also used to mitigate overfitting by achieving high robustness [32]. In fact, generating adversarial samples by slightly perturbing training samples may improve the generalization of deep neural models.

In this study, we use the popular state-of-the-art Fast Gradient Sign Method (FGSM) [33] to generate adversarial images. It is a white-box gradient-based algorithm that finds the loss to apply to an input image, in order to make decisions of a pre-trained neural model less overfitted on a specific class. The pre-trained model is the ViT architecture described above with parameters initially estimated on the original labeled images of $\mathcal{I}$.

The FGSM algorithm is based on the gradient formula:

$$g(\mathbf{I}) = \nabla_{\mathbf{I}} J(\theta, \mathbf{I}, y), \tag{4}$$

where $\nabla_{\mathbf{I}}$ denotes the gradient computed with respect to the imagery sample $\mathbf{x}$, and $J(\theta, \mathbf{I}, y)$ denotes the loss function of the ViT neural model initially trained on the original training set $\mathcal{I}$. In theory, FGSM determines the minimum perturbation ϵ to add to a training image $\mathbf{I}$ to create an adversarial sample that maximizes the loss function. According to this theory, given an input perturbation value ϵ, for each labeled image $(\mathbf{I}, y) \in \mathcal{I}$, a new image $(\mathbf{I}_{adv}, y) \in \mathcal{I}_{adv}$ can be generated such that:

$$\mathbf{I}_{adv} = \mathbf{I} + \epsilon \cdot sign(g(\mathbf{I})). \tag{5}$$

As $\mathcal{I}_{adv}$ is generated, parameters of the ViT architecture are finally estimated from the adversarially-augmented training set $\mathcal{I} \cup \mathcal{I}_{adv}$.

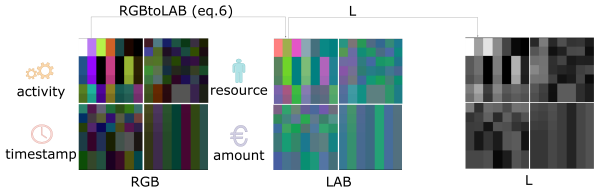

Figure 7: Map of attention produced in the RGB space (left side) for the prediction of the next-activity of the imagery representation shown in Figure 5 of the sample prefix trace reported in Figure 2. Transformation of the map of attention from the RGB space to the LAB space (central side). Luminosity (L) channel of the map of attention produced in the LAB space (right side).

4.3 Extracting maps of attention

Once the ViT parameters have been estimated, the ViT model is used to decide on the next-activity of any prefix trace. As described in [11], the Attention Rollout method [34] is used to extract the map of attention of the decision of the ViT model on a single sample. The Rollout method computes the map of attention by averaging the attention weights of the ViT model across all heads and then recursively multiplying the weight matrices of all layers. This accounts for the mixing of attention across patches through all layers. Then, we derive a quantitative indicator of the importance of events within patches by exploiting the lightness information of attention maps. The lighter the pixel in the attention map, the higher the effect of the pixel information enclosed in the image of the prefix trace on the ViT decision. Indeed, the generated attention maps are represented in the RGB color space, which operates on three channels (red, green, and blue) and does not provide information about lightness. Hence we transform the RGB representation of attention maps into the LAB color space, which operates on three different channels: the color lightness (L), the color ranges from green to red (A), and the color ranges from blue to yellow (B).

The transformation from the RGB space to the LAB space is performed as follows [35]:

$$\begin{aligned} L &= 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B, \\ A &= 1.4749(0.2213 \cdot R - 0.3390 \cdot G + 0.1177 \cdot B) + 128, \\ B &= 0.6245(0.1949 \cdot R + 0.6057 \cdot G - 0.8006 \cdot B) + 128. \end{aligned} \quad (6)$$

As an example, Figure 7 shows the map of attention of the next-activity decision produced by the ViT model for the imagery representation shown in Figure 5 for the prefix trace reported in Figure 2. The ViT model was learned through adversarial training from the training set of BPI12WC in the experimental study. The map of attention shown in Figure 7 (left side) is extracted in the RGB space with the roll-out method. Figure 7 (central side) shows the transformation of the map of attention from the RGB space to the LAB space completed according to Eq. 6. Finally, Figure 7 (right side) shows the luminosity values of the map of attention that indicate the influence of the patches on the prediction.

Table 2: Event log description: number of traces (♯**Trace**), number of events (♯**Event**), and set of views (**View**). For categorical views, the number of distinct categories in the view is reported in brackets.

Event log	♯Trace	♯Event	View
BPI12W	9658	170107	activity (19), timestamp, resource (60), loan amount
BPI12WC	9658	72413	activity (6), timestamp, resource (60), loan amount
BPI12C	13087	164506	activity (23), timestamp, resource (69), loan amount
BPI13I	7554	65533	activity (13), timestamp, resource (1440), impact (4), org group (649),org role (24), org country (23), org involved (25), product (704), resource country (32)
BPI13P	2306	9011	activity (7), timestamp, resource (643),impact (4), org group (15), org role (29), org country (18), product (380), resource country (21)
Receipt	1434	8577	activity (27), timestamp, resource (48),channel (5), department (3), group (8), org group (10), responsible (39)
BPI17O	42995	193849	activity (8), timestamp, resource (144),monthly cost, first withdrawal amount, credit score, offered amount, number of terms, action (2)
BPI20R	6886	36796	activity (19), timestamp, resource (2), org (36), project (79), task (597), role (8)

5 Experimental Set-up

In this section, we describe the event logs used for evaluating the accuracy and explainability of JARVIS, the experimental set-up and the implementation details.

5.1 Event logs

We processed eighth real-life event logs available on the 4TU Centre for Research.[1] BPI12W, BPI12WC and BPI12C were provided for the BPI Challenge 2012 [36]. These logs contain events collected monitoring the loan application process of a Dutch financial institute. They record: activities that track the state of the application (A), activities that track the state of work items associated with the application (W), and activities that track the state of the offer (O). Sub-processes A and O contain only the complete life cycle transition, while sub-process W includes scheduled started and completed life cycle transitions. In particular, BPI2012C contains all the traces, but retains the completed events of such traces, BPI2012W contains traces of sub-process W, and BPI2012WC contains traces of sub-process W, but retains the completed events of such traces. BPI13I and BPI2013P [37] contain events collected from the closed problem management system and the incident management system, respectively, of Volvo IT in Belgium. Receipt [38] was collected in the CoSeLoG project. It records events produced in an anonymous municipality in the Netherlands during the receiving phase of the building permit application process. BPI17O [39] contains events regarding all

1. https://data.4tu.nl/portal

offers made for an accepted application through an online system of a Dutch financial institute in 2016 and their subsequent events until February 1st 2017, 15:11. BPI20R [40] contains events pertaining requests for Payment, which are not travel-related. In 2017, events were collected for two departments, in 2018 events were collected for the entire university. A summary of the characteristics of these event logs is reported in Table 2.

5.2 Experimental set-up

The experimental setting described in [1] was adopted in this study. A temporal split was performed to divide each event log into train and test traces. To this aim, the traces of an event log were sorted by the starting timestamp. The first 67% were selected for training the predictive model, while the remaining 33% were considered to evaluate the performance of the learned model on the unseen traces.

5.3 Implementation details

JARVIS was implemented in Python 3.9.12 – 64 bit version – using Keras 2.8.0 library that is a high-level neural network API using TensorFlow 2.8.1 as the back-end.[2] The ViT architecture was trained with mini-batches using back-propagation. The tree-structured Parzen estimator [41] was used to perform the automatic hyper-parameter optimization of the ViT architecture. This allowed us to automate the decisions related to the parameters of the implementation of the ViT based on the characteristics of the training data. The optimization phase was performed by exploring both the number of layers L and the number of Heads H in $\{2, 3, 4\}$, latent vector size D in [2^5,2^8], MLP size in [2^4,2^6], batch size in [2^6,2^8], and learning rate in [10^{-3},10^{-2}]. It was conducted by using the 20% of the training set, selected with stratified sampling, as a validation set, according to the Pareto Principle. The hyper-parameter configuration, which minimized the loss on the validation set in the range of possible values explored in the defined search space, was automatically selected for each event log. The gradient-based optimization was performed using the Adam update rule to optimize the categorical cross-entropy loss function. The maximum number of epochs was set equal to 200 and an early stopping approach was used to avoid overfitting (as in [3]). The training phase stopped when there was no improvement of the loss on the validation set for 10 consecutive epochs. For the early stopping, we used the same validation set considered in the hyper-parameter optimization phase.

The adversarial image generation was performed using the algorithm FGSM (as implemented in the Adversarial Robustness Toolbox 1.12.1 library[3]) with the perturbation value ϵ set equal to 0.001 by default. FGSM is one of the most popular adversarial sample generators that is prone to catastrophic overfitting [32]. In this study, we adopted the default set-up $\epsilon = 0.001$. This decision was based on the study of [12] that suggests setting perturbation ϵ as a small value in the range between 0 and 0.1, to scale the noise and ensure that perturbations are small enough to remain undetected to the human eye, but large enough to help in generalizing the neural model.

2. https://github.com/vinspdb/JARVIS
3. https://adversarial-robustness-toolbox.readthedocs.io/

6 Accuracy performance analysis

In this section we show the results of an analysis aimed at evaluating the accuracy of JARVIS, to answer the following research questions:

- Q1 How does the multi-view, ViT-based learning schema compare to state-of-the-art PPM methods?
- Q2 Is the adversarial training strategy able to achieve higher accuracy than non-adversarially trained ViT counterpart on unseen traces?
- Q3 Is the adversarial training strategy robust to the selection of the algorithm for the generation of adversarial samples?
- Q4 How does the accuracy of the ViT model change by modifying the order according to multiple views assigned to image patch positions?

To answer Q1 we performed a comparative study involving several deep learning methods based on LSTM, CNN and Transformer architectures, which were selected from the recent PPM literature (i.e., [3], [4], [9], [20], [23]). These methods were run with the information enclosed in all views recorded in the study event logs for a safe comparison. This analysis allows us to explore how the proposed imagery representation and the use of a ViT model can actually aid in gaining accuracy in next-activity prediction problems. The results of this analysis are illustrated in Section 6.1. To answer Q2 we performed an ablation study comparing the accuracy performance of JARVIS to that of its baseline obtained by keeping away the generation of adversarial images and completing the training of the ViT model with images of prefix traces actually observed in the training set only. This analysis explores how adversarial training can actually contribute to gaining accuracy in JARVIS. The results of this analysis are illustrated in Section 6.2. To answer Q3 we performed a sensitivity study exploring the performance of JARVIS achieved by using different, state-of-the-art algorithms (i.e., FGSM, PGD and DeepFool) to generate adversarial samples. This analysis aims to explore the validity of our decision to use FGSM as default adversarial sample generation algorithm. The results of this analysis are illustrated in Section 6.3. To answer Q4 we performed an experiment randomly shuffling the positions of views in the event logs to explore how the performance of JARVIS is robust to the view order. The results of this analysis are illustrated in Section 6.4.

For each event log, we learned the next-activity predictive hypothesis functions of the compared methods on the training set and evaluated their ability to predict the next activity on the running traces of the testing set. We analyze the macro FScore and the macro GMean performances achieved. Both the macro FScore and the macro GMean are well-known multi-class classification metrics commonly used in imbalanced domains. They measure the average FScore and GMean per activity type i giving equal weights to each activity type. In this way, we avoid that our evaluation offsets the possible impact of imbalanced data learning. The FScore of activity i measures the harmonic mean of precision and recall of i, i.e., $FScore_i = 2\frac{precision_i \times recall_i}{precision_i + recall_i}$ so that $FScore = \frac{1}{k}\sum_{i=1}^{k} 2\frac{precision_i \times recall_i}{precision_i + recall_i}$. The GMean

of activity i measures the geometric mean of specificity and recall of i by equally considering the errors on opposite classes, i.e., $GMean_i = \sqrt{specificity_i \times recall_i}$ so that $GMean = \frac{1}{k}\sum_{i=1}^{k}\sqrt{specificity_i \times recall_i}$. The higher the macro FScore and the macro GMean, the better the accuracy performance of the method.

6.1 Related method analysis

We compared the performance of JARVIS to that of the methods described in [3], [4], [9], [20] and [23]. [3] learns a LSTM model from a sequence representation of prefix traces. [4] learns a CNN with Inception from RGB images of prefix traces. [9] learns a LSTM model with Attention from a sequence representation of prefix traces. [20] learns a CNN model from gray-scale images of prefix traces. [23] learns a Transformer model from a sequence representation of prefix traces. All related methods, except for [3], were originally experimented by their authors accounting for specific views of traces. Specifically, [4] and [9] were experimented with activity, resource, and timestamp information, [20] was experimented with activity and timestamp information, and [23] was experimented with activity information. To provide a fair comparison, we ran these related methods by accounting for all views recorded in the considered event logs. In fact, as the authors of the considered related methods made the code available, we were able to run all the compared algorithms in the same experimental setting, thus performing a safe comparison. Parameters of the related methods were set according to the best parameter set-up determined in the code provided by the authors and described in the code repositories. Notice that this comparative study was performed by selecting [3] from the group of related methods (i.e., [1], [2], [3], and [16]) reported in Table 1, which train a LSTM model for next-activity prediction from sequence data. In fact, [3] generalizes [1], [2], [16] as it was originally formulated to account for all views recorded in an event log.

Table 3 collects the macro FScore and the macro GMean of both the considered related methods and JARVIS. These results deserve several considerations. Notably, JARVIS achieves the highest FScore and GMean in five out of eight event logs, being the runner-up method in one out of eight event logs. In addition, JARVIS always outperforms the two related methods using an imagery encoding strategy [4], [20] except for BPI12W. Specifically, it always outperforms the related method using a Transformer [23]. It commonly outperforms the related method using the attention modules [9] except for the macro FScore in BPI12W, and both macro FScore and macro GMean in BPI13I. These conclusions are also drawn from the critical difference diagram reported in Figure 8 for the macro FScore performance of the compared methods. This diagram was obtained after rejecting the null hypothesis with *p-value* ≤ 0.05 in the Friedman's test and adopting the post-hoc Nemenyi test for pairwise method comparisons. Despite there is no statistical difference between JARVIS, [3], [4], [9] and [23], JARVIS is the top-ranked in the difference diagram with [3] as runner-up. This result assesses the effectiveness of our idea of using a ViT model, trained from a multi-view imagery representation of prefix traces, in PPM problems of next-activity prediction.

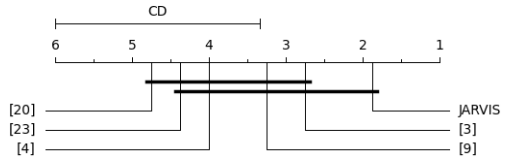

Figure 8: Comparison of FScore of JARVIS and the state-of-the-art methods defined in [3], [4], [9], [20] and [23] with the Nemenyi test. Groups of approaches that are not significantly different (at $p \leq 0.05$) are connected.

6.2 Adversarial training strategy analysis

To evaluate the effect of adversarial training we compare the performance of JARVIS to that of its non-adversarial counterpart JARVIS$^{\ominus ADV}$. JARVIS$^{\ominus ADV}$ keeps away the adversarial training strategy and returns the ViT model whose parameters were estimated on the original training set.

Table 4 collects the macro FScore and macro GMean of both JARVIS and JARVIS$^{\ominus ADV}$. These results show that the use of the adversarial training strategy allows us to higher values of either macro FScore or macro GMean in the event logs of this study with the exception of BPIC12C. This is the only event log, where JARVIS$^{\ominus ADV}$ achieves higher values of both macro FScore and macro GMean than JARVIS.

To examine in depth the effect of the adversarial training strategy on single classes we explore the FScore computed per activity. As an example, Table 5 collects the FScore computed per activity for JARVIS and JARVIS$^{\ominus ADV}$ in both BPI13P and BPI20R. These two event logs correspond to two scenarios with a small number of distinct activities (7 activities in BPI13P) and a high number of distinct activities (19 activities in BPI20R), respectively. Both event logs include several imbalanced activities (e.g., a1 and a3 in BPI13P, as well as a6, a11 and a13 in BPI20R). The use of the adversarial training strategy increases accuracy on the majority activities for BPI20R (e.g., a2, a8), as well as in minority activities (e.g., a1, a3) for BPI13P). So, despite the use of the adversarial training strategy does not provide a systematic solution to the imbalanced phenomenon, it may generally help in mitigating overfitting on majority activities and gaining accuracy on some minority activities.

6.3 Adversarial sample generation algorithm analysis

We analyze the performance of three white-box adversarial sample generation algorithms, that is, FGSM [33], PGD [42] and DeepFool [43]. FGSM is used in the rest of this experimental study. PGD performs an iterative version of FGSM. DeepFool performs an iterative procedure to find the minimum adversarial perturbations on both an affine binary classifier and a general binary differentiable classifier. It integrates the one-versus-all strategy to be applied to multi-class problems. Both PGD and DeepFool commonly spend more training time than FGSM since they perform multiple trials to generate perturbations. These methods are considered to be among the state-of-the art methods for adversarial training in the image domains [12].

Table 6 shows the macro FScore and the macro GMean achieved by JARVIS by varying the adversarial sample generation algorithm among FGSM (baseline), PGD and

Table 3: Comparison between JARVIS and related methods defined in [3], [4], [9], [20] and [23] : macro FScore and macro GMean. The best results are in bold, while the runner-up results are underlined.

Eventlog	FScore						GMean					
	JARVIS	[3]	[4]	[9]	[20]	[23]	JARVIS	[3]	[4]	[9]	[20]	[23]
BPI12W	0.667	**0.737**	0.692	0.673	0.673	0.661	0.820	**0.847**	0.828	0.792	0.819	0.825
BPI12WC	**0.705**	0.685	0.661	0.675	0.645	0.668	**0.812**	0.798	0.778	0.792	0.780	0.787
BPI12C	0.644	**0.654**	0.642	0.638	0.643	0.624	0.786	**0.792**	0.782	0.785	0.781	0.781
BPI13P	**0.414**	0.320	0.336	0.408	0.228	0.405	**0.595**	0.533	0.546	0.594	0.472	0.593
BPI13I	0.387	0.405	0.295	**0.407**	0.363	0.380	0.615	**0.626**	0.534	**0.626**	0.594	0.603
Receipt	**0.525**	0.455	0.409	0.471	0.302	0.383	**0.733**	0.676	0.646	0.702	0.563	0.620
BPI17O	**0.720**	0.714	0.705	0.691	0.718	0.712	**0.846**	0.833	0.830	0.815	0.835	0.831
BPI20R	**0.491**	0.450	0.483	0.455	0.432	0.481	**0.699**	0.660	0.691	0.664	0.643	0.683

Table 4: JARVIS vs JARVIS$^{\ominus ADV}$: macro FScore and macro GMean. The best results are in bold.

Eventlog	FScore		GMean	
	JARVIS	JARVIS$^{\ominus ADV}$	JARVIS	JARVIS$^{\ominus ADV}$
BPI12W	**0.667**	0.628	**0.820**	0.799
BPI12WC	**0.705**	0.689	0.812	**0.816**
BPI12C	0.644	**0.656**	0.786	**0.793**
BPI13P	**0.414**	0.405	**0.595**	0.592
BPI13I	**0.387**	0.344	**0.615**	0.572
Receipt	**0.525**	0.490	**0.733**	0.712
BPI17O	**0.720**	0.716	0.846	**0.849**
BPI20R	**0.491**	0.482	**0.699**	0.688

Table 5: FScore per activity of JARVIS and JARVIS$^{\ominus ADV}$ in BPI13P and BPI20R. The best results are in bold.

Event log	Activity	JARVIS	JARVIS$^{\ominus ADV}$	Support%
BPI13P	a1	**0.337**	0.315	10.80%
	a2	0.665	**0.705**	29.80%
	a3	**0.234**	0.120	9.34%
	a4	0.456	**0.486**	27.64%
	a5	0.375	**0.402**	22.42%
BPI20R	a1	**0.998**	**0.998**	20.64%
	a2	**0.957**	0.851	21.20%
	a3	**0.077**	0.007	7.68%
	a4	0.000	0.000	0.00%
	a5	0.000	0.000	0.13%
	a6	0.000	0,000	0.01%
	a7	0.966	**1.000**	0.13%
	a8	**0.827**	0.795	20.51%
	a9	0.000	0.000	0.01%
	a10	0.603	**0.641**	3.02%
	a11	0.000	0.000	0.13%
	a12	**0.975**	**0.975**	3.39%
	a13	0.000	0.000	0.39%
	a14	**0.961**	**0.961**	2.13%
	a15	**0.998**	**0.998**	20.63%

DeepFool. Results show that FGSM outperforms PGD and DeepFool in five out of eights event logs. DeepFool commonly outperforms FGSM and PGD in the remaining three event logs (i.e., BPI12C, BPI13P and BPI20R). The only exception is observed with the macro GMean of BPI13P that achieves the highest value with PGD. Let us focus the attention on the performance of DeepFool and FGSM in BPI12C, BPI13P and BPI20R. JARVIS with DeepFool

Table 6: Adversarial sample generation: macro FScore and macro GMean of JARVIS by using FGSM (baseline), PGD and DeepFool

Eventlog	FScore			GMean		
	FGSM	PGD	DeepFool	FGSM	PGD	DeepFool
BPI12W	**0.667**	0.608	0.562	**0.820**	0.793	0.771
BPI12WC	**0.705**	0.692	0.700	**0.812**	0.807	0.809
BPI12C	0.644	0.647	**0.657**	0.786	0.784	**0.791**
BPI13P	0.414	0.417	**0.422**	0.595	**0.600**	0.599
BPI13I	**0.387**	0.342	0.328	**0.615**	0.576	0.568
Receipt	**0.525**	0.498	0.477	**0.733**	0.720	0.677
BPI17O	**0.720**	0.712	0.711	**0.846**	0.840	0.837
BPI20R	0.491	0.498	**0.525**	0.699	0.695	**0.718**

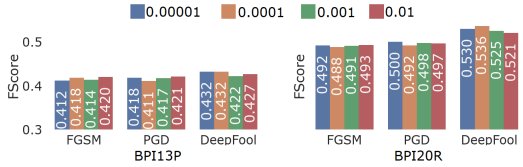

Figure 9: Macro FScore measured in BPI13P and BPI20R by varying the ϵ among 0.01, 0.001 (default), 0.0001 and 0.00001 with FGSM, PGD and DeepFool

achieves higher macro FScore than [3] in BPI12C, while JARVIS with FGSM achieves lower macro FScore than [3] in the same log (see results in Table 3). On the other hand, JARVIS with FGSM already outperforms all related methods reported in Table 3 in both BPI13P and BPI20R. So, FGSM can be considered a good choice to allow JARVIS to achieve higher accuracy than related methods.

Both FGSM, PGD and DeepFool take ϵ as input parameter. Figure 9 reports the macro FScore of JARVIS run with FGSM, PGD and DeepFool by varying ϵ among 0.01, 0.001 (default), 0.0001 and 0.00001 in both BPI13P and BPI20R. In both experiments, differences in macro FScore are negligible.

6.4 Image patch position analysis

We explore the effect of the single-view image patch positions within the adopted imagery encoding schema. We

Table 7: Image patch position analysis: macro FScore and macro GMean of JARVIS measured in BPI13P and BPI20R by processing views in the default order they appear in the event log (Original). Average and Standard deviation of the macro FScore and the macro GMean of JARVIS by processing views after performing the random shuffle of views. The shuffle operation is repeated on five trials.

Eventlog	Fscore		GMean	
	Original	Shuffle (Avg±Dev)	Original	Shuffle (Avg±Dev)
BPI13P	0.415	0.413±0.018	0.595	0.600±0.010
BPI20R	0.491	0.491±0.008	0.699	0.695±0.003

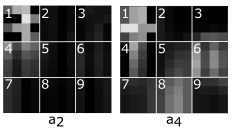

Figure 10: Maps of attention of two prefix traces of BPI13P, which are correctly labeled with "a2" ("Accepted-In Progress") and "a4" ("Completed-Closed"), respectively. The maps are shown in the luminosity channel of the LAB space. The numbers identify the names of the views in the event log: 1-"activity", 2-"resource", 3-"timestamp", 4-"impact", 5-"org country", 6-"org group, 7-"org role", 8-"product", 9-"resource country".

recall that, in this study, event log views are processed in the same order they are recorded in the event log, while the single view-based patches of each prefix trace are distributed in the prefix trace imagery grid from left to right, and from top to bottom. In this section, we evaluate the possible effect of the image patch positions on the accuracy of the ViT model trained by JARVIS. Let O be the ViT model that was trained by processing event log views in the order they were recorded in the event log. We trained five ViT models after randomly shuffling the views' positions in the event logs and, consequently, the positions of the view-based image patches in the imagery encoding of the prefix traces.

Table 7 shows the average and standard deviation of the macro FScore and the macro GMean of JARVIS, in both BPI13P and BPI20R, measured on the five trials of the view shuffle operation, as well as the macro FScore and the macro GMean achieved processing the views in the original order. The differences in the performances of the trained ViT models are negligible. This shows that the accuracy of the proposed approach is robust to the view order and the adopted ViT model can be trained by taking advantage of the intra-view information and inter-view relationships independently of the view positions in image patches.

7 Explanation Analysis

This analysis aimed to explore how intrinsic explanations enclosed in the attention maps generated through the ViT model may provide useful insights to explain model decisions. We intend to answer the following research questions:

- Q1 What is the effect of information enclosed in different view-based image patches on decisions?
- Q2 What is the effect of events within each view-based image patch on decisions?

For this purpose, in the following of this study, we explore several average measurements of the lightness information enclosed in the maps of attention, which were extracted for the training prefix traces of the event logs considered in this study. In particular, let $\mathbf{I}$ be the multi-patch imagery representation of a prefix trace. Let $\mathbf{L}$ be the lightness channel of the map of the attention of the ViT model on $\mathbf{I}$. For each view, we can compute the local patch lightness of the considered view-based patch in $\mathbf{L}$ as the average of the lightness measured on all pixels of $\mathbf{L}$ falling in the patch under study. This local measurement of the patch lightness quantifies the local importance of the information enclosed in the considered view-based image patch on the sample decision. The analysis of local patch lightness allows us to answer Q1. The highest the value of the local patch lightness, the highest the effect of the corresponding view on the sample decision. On the other hand, for each event positioned in a prefix trace (e.g., first event, second event), and for each view, we compute the local intra-patch event lightness as the average of the lightness associated with the pixels encoding the event in the view-based patch. The analysis of local intra-patch event lightness allows us to answer Q2. The highest the lightness of the local intra-patch event lightness, the highest the effect of the view information of the selected event on the sample decision.

We start analyzing explanations of local decisions concerning the next activity predicted for specific prefix traces. For example, Figure 10 shows the lightness channel of the attention maps extracted from the ViT model trained by JARVIS on two prefix traces of BPI13P. These prefix traces were correctly labeled with the next-activity "a2" ("Accepted-In Progress") and "a4" ("Completed-Closed"), respectively. Table 8 reports the local patch lightness measured for each view in the maps of attention shown in Figure 10. These results show that the patch associated with "activity" conveys the most relevant information for recognizing both "Accepted-In Progress" and "Completed-Closed" as the next-activities of the two sample prefix traces. However, "impact" and "org group" are the second and third most important views for the decision on the next-activity "Accepted-In Progress", while "org group" and "product" are the second and third most important views for the decision on the next-activity "Completed-Closed". Notably, "product", which is one of the top-three ranked views for the decision on the next activity "Completed-Closed", is the less important view for the decision on the next activity "Accepted-In Progress". This analysis shows that different views may convey the most important information for different decisions.

We go deeper in the explanation of the decision "Completed-Closed". Figure 11 shows the sequences of activities, org groups, and products recorded in the sample prefix trace with next-activity "Completed-Closed". We selected these views as they are the top-three views for recognizing the next activity "Completed-Closed" in the sample prefix trace according to the analysis of the local patch analysis reported in Table 8. For each selected view, each event of the prefix trace is annotated with the local

Table 8: Patch lightness measured for each view of BPI13P in the two maps of attention shown in Figure 10

View	Left map ("a2")	Right map ("a4")
activity	91.56	120.00
resource	10.50	11.88
timestamp	15.81	19.43
impact	69.56	45.00
org country	12.19	39.81
org group	23.06	100.25
org role	20.94	36.31
product	5.94	95.81
resource country	13.88	28.38

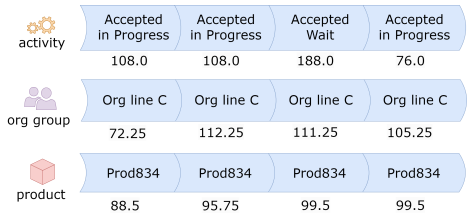

Figure 11: Intra-patch event lightness of each event of the sample prefix trace with the next-activity "a4" ("Completed-Closed"). The intra-patch event lightness is computed for the three-top patches in the map of attention shown in the right side of Figure 10 ("activity", "org group" and "product") selected according to the patch lightness.

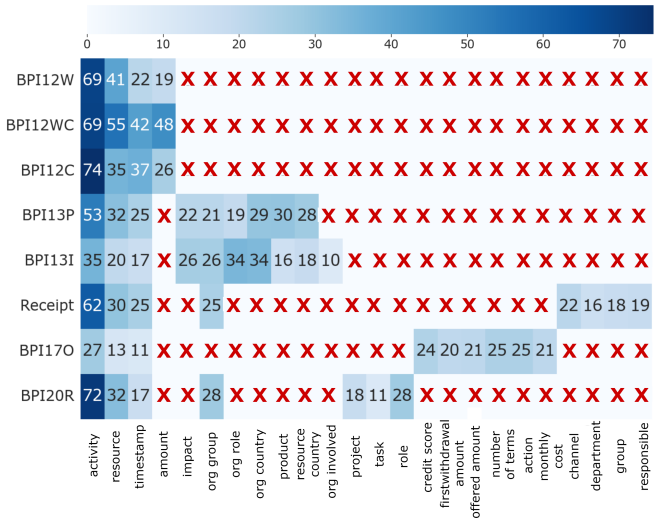

Figure 12: Heatmap of the global patch lightness computed for all event log views (axis X) in all event logs (axis Y). "×" denotes that the view reported on the axis X is missing in the event log reported on the axis Y.

intra-patch event lightness computed for the event within the patch of the attention map of the prefix trace, which is associated with the view. This plot explains that the activity "Accepted Wait" recorded in the third event of the prefix trace, the org group "Org line C" recorded in the second event of the prefix trace and the product "Prod84" recorded in the third and fourth positions of the prefix trace have the highest effect on the decision of predicting "Completed-Closed"' as next-activity of this sample prefix trace.

We continue analyzing the global effect of different

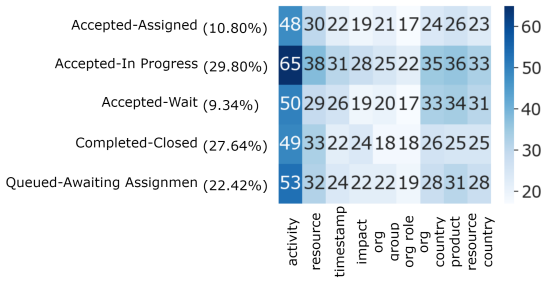

Figure 13: Heatmap of the global patch lightness computed activity by activity (axis Y) for all the views in the event log BPI13P. For each (next) activity category, the support of the category is reported in the brackets.

views by accounting for the patch lightness computed for each view and averaged on all the prefix traces of the training set. Figure 12 shows the heatmap of the average patch lightness computed on the training set in the event logs of this study. This map shows which views have the higher global effect on the ViT decisions. As expected, the activity information is globally the most important information for the ViT decisions in all the event logs. However, this explanation information shows that the "product" information is globally in the top-three ranked views in BPI13P, while "number of terms" and "action" information are globally in the top-three ranked views in BPI17O. These results support the decision of designing a multi-view approach not necessarily based on the standard views (activity, timestamp and resource). In fact, this analysis shows that changing the process changes the type of information most useful for predicting the next activity of every running trace of the study process. So, designing multi-view methods, which are able to incorporate any view, can actually contribute to gaining accuracy in next-activity prediction models.

In addition, to examine in depth the global effect of different views on different categories of next activities, we analyze the patch lightness, computed for each view, and averaged on each collection of training prefixes labeled with a category of next activity. Figure 13 shows the average of the patch lightness computed on every next activity category appearing in the training set of the event log BPIC13P. These results show that the effect of specific views on the ViT decisions may change depending on the next activity category. In particular, "activity" conveys the most relevant information for all activities. Instead "resource", that is the second most important view for the next-activity "a1" ("Accepted-Assigned"), "a2" ("Accepted-In Progress"), "a4" ("Completed-Closed") and "a5" ("Queued-Awaiting Assignment"), is the fifth most important view for the next activity "a3" ("Accepted-Wait"). On the other hand, "product" and "org country" are the second and third most important views for the next activity "a3" ("Accepted-Wait"). "timestamp" is never in the top-three ranked views for any next activity. This analysis shows that views may contribute differently to decisions about different types of activities within the same event log. The views that most contribute to recognizing a specific activity may change with respect to the activity to be recognized. Hence, this analysis

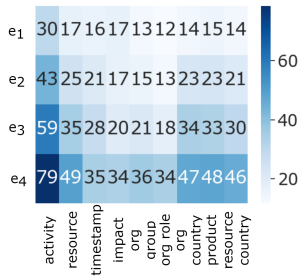

Figure 14: Heatmap of the global intra-patch event lightness computed for each event position (axis Y) in the prefix traces the event log BPI13P for each event log view (axis X)

supports the effectiveness of the decision to train the next-activity predictive model accounting for all views available in the event log. In fact, information enclosed in different views may contribute to gaining accuracy in recognizing different categories of activities within the same problem.

Finally, to explore the global effect of the intra-view, event information on ViT decisions, we analyze the average, intra-patch event lightness computed on the pixels of each studied patch, which encode the events occupying the position under study in a prefix trace. Figure 14 shows the average measurement of the intra-patch event lightness computed for each of the most recent four events recorded in the prefix traces of the training set of BPI13P. These results show that the most recent events recorded in a prefix conceive the most important information for the decisions. This conclusion can be drawn independently of both the category of the next activity and the view of the information considered in the event.

8 Conclusion

This paper illustrates a novel, multi-view, PPM method for next-activity prediction. We resort to an imagery representation that encodes multi-view information of a prefix trace as multiple color patches of an image. We take advantage of the self-attention modules of a ViT architecture to assign pairwise attention values to pairs of image patches being able to account for multi-view relationships. In addition, self-attention modules allow us to incorporate explainability directly into the structure of a PPM model by disclosing explainable information concerning specific views of the event log and events of the prefix trace that more influenced decisions. The experiments performed on several event logs show the accuracy of the proposed approach and explore the explanations produced through the attention mechanism.

Acknowledgments

Vincenzo Pasquadibisceglie, Giovanna Castellano and Donato Malerba are partially supported by the project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007), under the NRRP MUR program funded by the NextGenerationEU. Annalisa Appice is partially supported by project SERICS (PE00000014) under the NRRP MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

[1] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in International Conference on Advanced Information Systems Engineering, CAISE 2017, ser. LNCS. Springer, 2017, pp. 477–492.

[2] M. Camargo, M. Dumas, and O. G. Rojas, "Learning accurate LSTM models of business processes," in International Conference on Business Process Management, BPM 2019, ser. LNCS, vol. 11675. Springer, 2019, pp. 286–302.

[3] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "A multi-view deep learning approach for predictive business process monitoring," IEEE Trans. Serv. Comput., vol. 15, no. 4, pp. 2382–2395, 2022.

[4] ——, "Predictive process mining meets computer vision," in Business Process Management Forum, BPM 2020, ser. LNBIP, vol. 392. Springer, 2020, pp. 176–192.

[5] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, and G. Modugno, "ORANGE: outcome-oriented predictive process monitoring based on image encoding and cnns," IEEE Access, vol. 8, pp. 184 073–184 086, 2020.

[6] F. Taymouri, M. L. Rosa, S. M. Erfani, Z. D. Bozorgi, and I. Verenich, "Predictive business process monitoring via generative adversarial nets: The case of next event prediction," in 18th Int. Conf. on Business Process Man., BPM 2020, ser. LNCS. Springer, 2020, pp. 237–256.

[7] N. Mehdiyev, J. Evermann, and P. Fettke, "A novel business process prediction model using a deep learning method," Business & Information Systems Engineering, vol. 62, p. 143–157, 2018.

[8] R. Galanti and et al, "Explainable predictive process monitoring," in 2nd Int. Conf. on Process Mining. IEEE, 2020, pp. 1–8.

[9] B. Wickramanayake, Z. He, C. Ouyang, C. Moreira, Y. Xu, and R. Sindhgatta, "Building interpretable models for business process prediction using shared and specialised attention mechanisms," Knowledge-Based Systems, vol. 248, pp. 1–22, 2022.

[10] R. Galanti, M. de Leoni, M. Monaro, N. Navarin, A. Marazzi, B. Di Stasi, and S. Maldera, "An explainable decision support system for predictive process analytics," Engineering Applications of Artificial Intelligence, vol. 120, p. 105904, 2023.

[11] A. Dosovitskiy and et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in 9th Int. Conf. on Learning Representations, ICLR 2021.

[12] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," in 30th International Joint Conference on Artificial Intelligence, IJCAI 2021, 2021, pp. 4312–4321.

[13] W. Zhao, S. Alwidian, and Q. H. Mahmoud, "Adversarial training methods for deep learning: A systematic review," Algorithms, vol. 15, no. 8, 2022.

[14] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," Information Fusion, vol. 38, pp. 43–54, 2017.

[15] I. M. Kamal, H. Bae, N. I. Utama, and C. Yulim, "Data pixelization for predicting completion time of events," Neurocomputing, vol. 374, pp. 64–76, 2020.

[16] J. Evermann, J.-R. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," Decision Support Systems, vol. 100, pp. 129 – 140, 2017.

[17] A. Cuzzocrea, F. Folino, M. Guarascio, and L. Pontieri, "A multi-view learning approach to the discovery of deviant process instances," in On the Move to Meaningful Internet Systems: OTM 2015 Conferences, ser. LNCS. Springer, 2015, pp. 146–165.

[18] ——, "A robust and versatile multi-view learning framework for the detection of deviant business process instances," Int. J. Cooperative Inf. Syst., vol. 25, no. 4, pp. 1–56, 2016.

[19] F. Folino, G. Folino, M. Guarascio, and L. Pontieri, "A multi-view ensemble of deep models for the detection of deviant process instances," in ECML-PKDD 2020. Springer, 2020, pp. 249–262.

[20] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Using convolutional neural networks for predictive process analytics," in 1st International Conference on Process Mining, ICPM 2019. IEEE, 2019, pp. 129–136.

[21] V. Pasquadibisceglie, G. Castellano, A. Appice, and D. Malerba, "FOX: a neuro-fuzzy model for process outcome prediction and explanation," in 3rd International Conference on Process Mining, ICPM 2021. IEEE, 2021, pp. 112–119.

[22] M. Harl, S. Weinzierl, M. Stierle, and M. Matzner, "Explainable predictive business process monitoring using gated graph neural networks," J. Decis. Syst., vol. 29, pp. 312–327, 2020.

[23] Z. A. Bukhsh, A. Saeed, and R. M. Dijkman, "Processtransformer: Predictive business process monitoring with transformer network," CoRR, vol. abs/2104.00721, 2021.

[24] P. De Koninck, S. vanden Broucke, and J. De Weerdt, "act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes," in 6th Int. Conf. on Business Process Management, BPM 2018, ser. LNCS. Springer, 2018, pp. 305–321.

[25] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Darwin: An online deep learning approach to handle concept drifts in predictive process monitoring," Engineering Applications of Artificial Intelligence, vol. 123, pp. 1–21, 2023.

[26] S. Barbon Junior, P. Ceravolo, E. Damiani, and G. Marques Tavares, "Evaluating trace encoding methods in process mining," in International Symposium: From Data to Models and Back, DataMod 2020, ser. LNCS. Springer, 2021, pp. 174–189.

[27] G. M. Tavares, R. S. Oyamada, S. Barbon, and P. Ceravolo, "Trace encoding in process mining: A survey and benchmarking," Eng. Applications of Artificial Intelligence, vol. 126, p. 107028, 2023.

[28] M. Velmurugan, C. Ouyang, C. Moreira, and R. Sindhgatta, "Evaluating stability of post-hoc explanations for business process predictions," in 19th Int. Conf. on Service-Oriented Computing, ICSOC 2021, ser. LNCS. Springer, 2021, pp. 49–64.

[29] W. Rizzi, C. D. Francescomarino, and F. M. Maggi, "Explainability in predictive process monitoring: When understanding helps improving," in Business Process Management Forum - BPM Forum 2020,, ser. LNBIP. Springer, 2020, pp. 141–158.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in 1st Int. Conf. on Learning Representations, ICLR 2013, 2013.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Adv. in neural information proc. systems, vol. 30, pp. 1–11, 2017.

[32] M. Andriushchenko and N. Flammarion, "Understanding and improving fast adversarial training," in Annual Conf. on Neural Information Proc. Systems, NeurIPS 2020, 2020, pp. 16 048–16 059.

[33] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in 3rd International Conference on Learning Representations, ICLR 2015, 2015, pp. 1–11.

[34] S. Abnar and W. H. Zuidema, "Quantifying attention flow in transformers," in 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020. Association for Computational Linguistics, 2020, pp. 4190–4197.

[35] N. Nader, F. E.-Z. EL-Gamal, and M. E. l, "Enhanced kinship verification analysis based on color and texture handcrafted techniques," Research Square, 2022.

[36] B. van Dongen, "BPI Challenge 2012, 4TU.Centre for Research Data, Dataset," 2012. [Online]. Available: https://data.4tu.nl/repository/uuid:3926db30-f712-4394-aebc-75976070e91f

[37] W. Steeman, "BPI Challenge 2013, Ghent University, Dataset," 2013. [Online]. Available: https://data.4tu.nl/repository/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07

[38] J. Buijs, "Flexible evolutionary algorithms for mining structured process models," Ph.D. dissertation, Department of Mathematics and Computer Science, 2014.

[39] B. van Dongen, "BPI Challenge 2017 – Offer Log, 4TU.Centre for Research Data, Dataset," 2017. [Online]. Available: https://data.4tu.nl/articles/dataset/BPI_Challenge_2017_-_Offer_log/12705737

[40] ——, "BPI Challenge 2020 – Request for Payment, 4TU.Centre for Research Data, Dataset," 2020. [Online]. Available: https://data.4tu.nl/articles/dataset/BPI_Challenge_2020_Request_For_Payment/12706886

[41] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in Annual Conference on Neural Information Processing Systems, NIPS 2011, 2011, pp. 2546–2554.

[42] A. Madry and et al., "Towards deep learning models resistant to adversarial attacks," in 6th Int. Conf. on Learning Representations, 2018, pp. 1–10.

[43] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2016. IEEE, 2016, pp. 2574–2582.

Vincenzo Pasquadibisceglie received a Ph.D. in Computer Science from the University of Bari He is a Researcher in the Department of Computer Science of the University of Bari Aldo Moro. His research activity concerns process mining, deep learning and data-centric AI. He was involved in a regional research project on process mining. He received 2019 IET's Vision and Imaging Award ICDP 2019. He is member of the IEEE Task Force on Process Mining.

Annalisa Appice is an Associate Professor at the Department of Computer Science, University of Bari Aldo Moro, Italy. Her research interests include data mining with data streams, cyber and event data. She published more than 180 papers in international journals and conferences. She was Program co-Chair of ECMLPKDD 2015, ISMIS 2017 and DS 2020. She was Journal Track co-Chair of ECMLPKDD 2021. She was co-chair of several editions of ML4PM. She is a member of the editorial board of MACH, DAMI, EAAI and JIIS. She is member of the IEEE Task Force on Process Mining.

Giovanna Castellano is an Associate Professor at the Department of Computer Science, University of Bari Aldo Moro, Italy, where she is the coordinator of the Computational Intelligence Lab. She is member of the IEEE Society, the EUSFLAT society and the INDAM-GNCS society. Her research interests are in the area of Computational Intelligence and Computer Vision. She has published more than 200 papers in international journals and conferences. She is Associate Editor of several international journals. She was General chair of IEEE-EAIS2020. She is a member of the IEEE Task Force on Explainable Fuzzy Systems.

Donato Malerba is a Full Professor in the Department of Computer Science at the University of Bari Aldo Moro in Italy. He has been responsible for the local research unit in several European and national projects. Moreover, he has served as the Program (co-)Chair for conferences like IEA-AIE 2005, ISMIS 2006, SEBD 2007, ECMLPKDD 2011, and as the General Chair of ALT/DS 2016 and IJCLR 2023. He is also a member of the editorial boards of several international journals. He is the scientific coordinator of Spoke 6 - Symbiotic AI within the Italian project "Future AI Research (FAIR)," funded by the NextGenerationEU initiative. His academic career includes the publication of over 350 papers in international journals and conferences. His research interests primarily revolve around machine learning and big data analytics. He participates to the IEEE Task Force on Process Mining.