Enhancing Next Activity Prediction with Adversarial Training of Vision Transformers

Vincenzo Pasquadibisceglie[1,2,*,†], Annalisa Appice[1,2], Giovanna Castellano[1,2] and Donato Malerba[1,2]

[1]*University of Bari Aldo Moro, Bari, Italy*

[2]*Consorzio Interuniversitario Nazionale per l'Informatica - CINI, Bari, Italy*

Abstract

Predicting the subsequent activity in the ongoing execution (trace) of a business process is a crucial task in Predictive Process Monitoring (PPM). This capability enables analysts to intervene proactively and prevent undesirable behaviors. This paper presents a PPM approach that integrates adversarial training with Vision Transformers (ViTs) to enhance the accuracy of predicting the next activity in a running process trace. This approach takes into account multi-view information that may be captured in a process trace, treating them as distinct patches of an image. Attention modules are employed to reveal explainable information about the different views of a business process and the trace events that could influence the prediction. Additionally, to mitigate overfitting and improve accuracy, we investigate the impact of adversarial ViT training. Experiments conducted on various benchmark event logs demonstrate the effectiveness of the proposed approach compared to several state-of-the-art PPM techniques. Notably, the explanations obtained through attention modules yield valuable insights.

Keywords

Predictive process monitoring, Next activity prediction, Deep learning, Multi-view learning, Adversarial training, Vision transformers, Attention, XAI, Computer vision

1. Introduction

Predictive Process Monitoring (PPM) is a burgeoning field focused on enhancing business process efficiency and effectiveness through predictive analytics. By analyzing historical raw event data, PPM systems can identify patterns and trends, providing valuable insights into the key factors contributing to process inefficiencies and bottlenecks.

The use of deep learning in predictive modeling has become increasingly popular in PPM systems, reflecting the broader trend of deep learning's success across various domains. Specifically, several deep neural networks, such as Long Short-Term Memory (LSTM) networks [1], [2], [3], [4], Convolutional Neural Networks (CNNs) [5], [6], Generative Adversarial Networks (GANs) [7], and Autoencoders [8], have recently contributed to improving the accuracy of PPM

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy

[*] Corresponding author.

vincenzo.pasquadibisceglieds@uniba.it (V. Pasquadibisceglie); annalisa.appice@uniba.it (A. Appice); giovanna.castellano@uniba.it (G. Castellano); donato.malerba@uniba.it (D. Malerba)

0000-0002-7273-3882 (V. Pasquadibisceglie); 0000-0001-9840-844X (A. Appice); 0000-0002-6489-8628 (G. Castellano); 0000-0001-8432-4608 (D. Malerba)

systems. This is due to their ability to learn accurate deep neural models, which in turn enable proactive and corrective actions to enhance process performance and mitigate risks.

While the primary focus of PPM systems remains on delivering accurate predictions of future states of running traces, there is a growing preference for predictive models that are easier to explain in PPM applications. Recent studies [9, 10, 11, 12] have explored the application of existing eXplainable AI (XAI) methods to elucidate opaque PPM models. However, the issue of model explainability in the context of deep learning-based PPM systems remains under-explored. In [13], we recently introduced a method called JARVIS [13] (Joining Adversarial tRaining with VISion transformers in next-activity prediction), which combines Vision Transformers (ViT) and Adversarial Training to achieve a balance between model accuracy and explainability. Specifically, the model's explainability is enhanced by the adoption of a ViT, a deep neural architecture comprising multiple self-attention layers. An attention layer in deep learning is a component that enables a neural network to concentrate on specific parts of the input data when making predictions or decisions. It is inspired by the human visual system, which can selectively focus on different parts of an image to understand it better. Therefore, multiple self-attentions layers can provide an explanation of the model's behavior in terms of the most informative inputs. Adversarial training [14] is also employed to improve accuracy by incorporating perturbed (i.e., adversarial) inputs into the training process, thereby mitigating overfitting and enhancing generalization.

The paper is organized as follows. Preliminary concepts are reported in Section 2, while the JARVIS approach is described in Section 3. The experimental setup and the results of the evaluation of the proposed approach are illustrated in Sections 4. Finally, Section 5 recalls the purpose of our research, draws conclusions, and illustrates possible future developments.

2. Preliminary concepts

A *trace* is a record of a business process that shows the stages of its execution through a sequence of events. An *event* is a complex entity characterized by two essential components: the activity and the timestamp (indicating when the activity occurred). Additionally, events may possess optional characteristics, such as the *resource* responsible for the activity or the *cost* involved in completing it. Consequently, each event is accompanied by two mandatory views, representing the activity and the timestamp, as well as m optional views corresponding to other event characteristics. Let $\mathcal{A}$ be the set of all activity names, $\mathcal{S}$ be the set of all trace identifiers, $\mathcal{T}$ be the set of all timestamps, and $\mathcal{V}_j$ be the set of all names in the j-th view, where $1 \leq j \leq m$.

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 $\pi_{\mathcal{V}_j}\colon \mathcal{E} \mapsto \mathcal{V}_j$ such that $\pi_{\mathcal{V}_j}(e) = v_j$, where $j = 1, \ldots, m$.

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, $\mathcal{P}_{\mathcal{V}} = \{\Pi_{\mathcal{V}}(\sigma^k), a_{k+1} | (\sigma^k, \pi(e_{k+1}))) \in \mathcal{P}\}$.*

Given the prefix σ^k of a longer trace σ for which we do not know the actions in the rest of the sequence $\langle e_{k+1}, e_{k+2}, \ldots, e_n \rangle$, we can resort to machine learning techniques to learn a function $H \colon \mathcal{A}^* \mapsto \mathcal{A}$, from a labeled prefix trace multiset $\mathcal{P}$, such that $H(\sigma^k)$ predicts the expected next-activity a_{k+1}. More specifically, we frame the *next-activity prediction* task as a multi-class classification problem.

In this study, we represent the labeled multiset $\mathcal{P}$ as a collection of color image patches that are given as input to a ViT architecture [15]. This approach allows us to leverage ViT's self-attention mechanism to capture complex relationships between different parts of the input data. Moreover, we can simultaneously enhance the model's explainability, as the self-attention mechanism enables the model to focus on the most informative inputs.

3. JARVIS

The main phases of the JARVIS approach are described in the following. Specifically, Section 3.1 illustrates how 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. Section 3.2 describes how $\mathcal{I}$ is fed into a ViT architecture that is trained with adversarial training to estimate parameters of a next-activity prediction function H. Finally, Section 3.3 illustrates the extraction of the attention maps.

3.1. 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.

According to the multi-view formulation introduced in Section 2, 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. 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 one by resorting to the equal-frequency discretization algorithm. The number of discretization bins of a numeric characteristic is set equal to the average number of distinct categories in the original categorical views of the event log. This ensures that the granularity of the discretized variables is consistent with that of the other original categorical variables. 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 $\mathcal{L}$.

Subsequently, the multiset $\mathcal{P}$ is created by extracting traces from $\mathcal{L}$ and labeling them with the next activity. As prefix traces in $\mathcal{L}$ may vary in length, we employ a combination of padding and windowing techniques to ensure uniformity in the length of the prefix traces in $\mathcal{P}$. 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 [16] is then used to transform the categorical representation of a prefix trace into a bidimensional, numeric embedding representation. 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 $AVG_\sigma \times AVG_\sigma$.

Finally, for each labeled prefix trace $(\sigma^k, a_{k+1}) \in \mathcal{P}$, the list of its multi-view, bidimensional, numeric 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 [5]. 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 $(\lceil\sqrt{m+2}\rceil \cdot AVG_\sigma) \times (\lceil\sqrt{m+2}\rceil \cdot AVG_\sigma) \times 3$.

The generated multi-patch images are labeled as the corresponding prefix traces and added to the labeled image multiset $\mathcal{I}$. The parameters of the ViT architecture are estimated through the adversarial training strategy.

3.2. Adversarial Training

In this study, we use the popular state-of-the-art Fast Gradient Sign Method (FGSM) [17] 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: $g(\mathbf{I}) = \nabla_{\mathbf{I}} J(\theta, \mathbf{I}, y)$, 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 $\mathbf{I}_{adv} = \mathbf{I} + \epsilon \cdot sign(g(\mathbf{I}))$.

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

3.3. 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. The Attention Rollout method [18] is used to extract the map of attention of the decision of the ViT model on a single sample. 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 [19]: $L = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B$, $A = 1.4749(0.2213 \cdot R - 0.3390 \cdot G + 0.1177 \cdot B) + 128$, and $B = 0.6245(0.1949 \cdot R + 0.6057 \cdot G - 0.8006 \cdot B) + 128$.

4. Experiments

Section 4.1 describes the event logs used for evaluating the accuracy and explainability of JARVIS and the experimental set-up. Section 4.2 describes the accuracy results, while Section 4.3 describes the explanation results.

4.1. Event logs and experimental set-up

We analyzed eight real-life event logs available on the 4TU Centre for Research.[1] For each event log we performed a temporal split, dividing the log into training and testing traces. To achieve this, we sorted the traces of each event log by their starting timestamps. The first two-thirds of the sorted traces were chosen for training the predictive model, while the remaining one-third was reserved for evaluating the model's performance on unseen data.

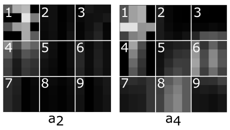
4.2. Accuracy performance analysis

We evaluated the performance of JARVIS against the methods outlined in [3], [5], [10], [20] and [21]. All methods, with the exception of [3], were initially tested by their respective authors, who considered specific views of traces. Specifically, [5] and [10] were experimented with activity, resource, and timestamp information, [20] was experimented with activity and timestamp information, and [21] 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. 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. Table 1 collects the macro FScore and the macro GMean of both the considered related methods and JARVIS. These results show that 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 [5, 20] except for BPI12W. Specifically, it always outperforms the related method using a Transformer [21]. It commonly outperforms the related method using the attention modules [10] except for the macro FScore in BPI12W, and both macro FScore and macro GMean in BPI13I.

Table 1
Comparison between JARVIS and related methods defined in [3], [5], [10], [20] and [21] : macro FScore and macro GMean. The best results are in bold, while the runner-up results are underlined.

Eventlog	FScore						GMean					
	JARVIS	[3]	[5]	[10]	[20]	[21]	JARVIS	[3]	[5]	[10]	[20]	[21]
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

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



(a) Maps of attention

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

(b) Patch lightness per view

Figure 1: (a) 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". (b) Patch lightness measured for each view of BPI13P in the two maps of attention.

4.3. 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. For example, Figure 1a 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. Figure 1b reports the local patch lightness measured for each view in the maps of attention shown in Figure 1a. 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". Significantly, the "product" view, which ranks among the top three for predicting the next activity "Completed-Closed", holds less importance when predicting the next activity "Accepted-In Progress". This analysis underscores the notion that distinct views may carry varying degrees of significance depending on the specific decision being made.

Finally, we examine the global effect of different views by accounting for the patch lightness computed for each view and averaged on all the prefix traces of the training set. Figure 2 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 ViT decisions. As expected, the activity information is globally the most important information for 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, whereas "number of terms" and "action" information are globally in the top-three ranked views in BPI17O. Findings lend support to the decision to develop a multi-view approach that does not solely rely on the standard views

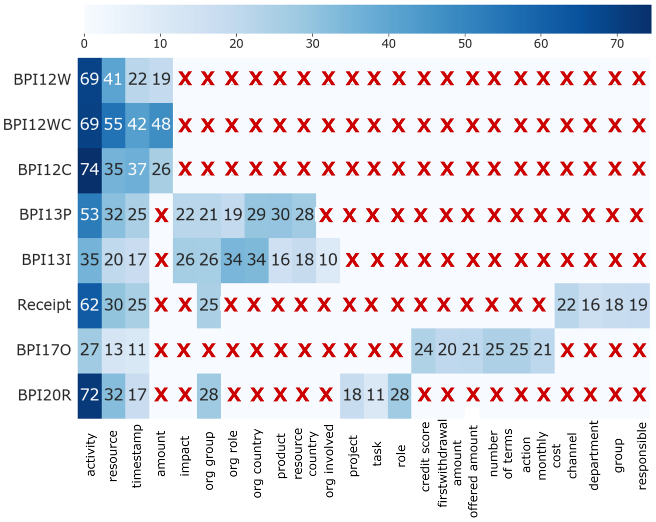


Figure 2: Heatmap of the global patch lightness computed for all views (axis X) in all event logs (axis Y). "×" denotes that the view reported on the axis X is missing in the event log reported on the axis Y.

(activity, timestamp and resource). They demonstrate that the type of information most valuable for predicting the next activity in each running trace may depend on the type of study process of which the trace is an execution.

5. Conclusion

This paper introduces a Predictive Process Monitoring (PPM) approach designed to forecast the subsequent activity in a sequence of events. The method employs an image-based representation of multiple views of the event sequence. It employs a ViT architecture, which utilizes self-attention modules to assign attention values to pairs of image patches, thereby capturing relationships between different views of the process data. Moreover, the self-attention modules allow for the integration of explainability into the model's structure by providing insights into specific views and events that influenced the predictions. The proposed approach is assessed using various event logs, and the results illustrate its accuracy and the advantages of the attention mechanism's explanatory capabilities.

6. 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, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: International Conference on Advanced Information Systems Engineering, CAISE 2017, LNCS, Springer, 2017, pp. 477–492.

[2] M. Camargo, M. Dumas, O. González-Rojas, Learning accurate lstm models of business processes, in: Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17, Springer, 2019, pp. 286–302.

[3] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, A multi-view deep learning approach for predictive business process monitoring, IEEE Transactions on Services Computing 15 (2022) 2382–2395. doi:10.1109/TSC.2021.3051771.

[4] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, Darwin: An online deep learning approach to handle concept drifts in predictive process monitoring, Engineering Applications of Artificial Intelligence 123 (2023) 106461. URL: https://www.sciencedirect.com/science/article/pii/S0952197623006450. doi:https://doi.org/10.1016/j.engappai.2023.106461.

[5] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, Predictive process mining meets computer vision, in: Business Process Management Forum, BPM 2020, volume 392 of *LNBIP*, Springer, 2020, pp. 176–192.

[6] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, G. Modugno, ORANGE: outcome-oriented predictive process monitoring based on image encoding and cnns, IEEE Access 8 (2020) 184073–184086. doi:10.1109/ACCESS.2020.3029323.

[7] F. Taymouri, M. L. Rosa, S. M. Erfani, Z. D. Bozorgi, 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, LNCS, Springer, 2020, pp. 237–256.

[8] N. Mehdiyev, J. Evermann, P. Fettke, A novel business process prediction model using a deep learning method, Business & Information Systems Engineering 62 (2018) 143–157. doi:10.1007/s12599-018-0551-3.

[9] R. Galanti, et al, Explainable predictive process monitoring, arXiv preprint arXiv:2008.01807 (2020).

[10] B. Wickramanayake, Z. He, C. Ouyang, C. Moreira, Y. Xu, R. Sindhgatta, Building interpretable models for business process prediction using shared and specialised attention mechanisms, Knowledge-Based Systems 248 (2022) 1–22. doi:https://doi.org/10.1016/j.knosys.2022.108773.

[11] R. Galanti, M. de Leoni, M. Monaro, N. Navarin, A. Marazzi, B. Di Stasi, S. Maldera, An explainable decision support system for predictive process analytics, Engineering Applications of Artificial Intelligence 120 (2023) 105904. doi:https://doi.org/10.1016/j.engappai.2023.105904.

[12] V. Pasquadibisceglie, A. Appice, G. Ieva, D. Malerba, Tsunami - an explainable ppm approach for customer churn prediction in evolving retail data environments, Journal of Intelligent Information Systems (2023). URL: https://doi.org/10.1007/s10844-023-00838-5. doi:10.1007/s10844-023-00838-5.

[13] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, Jarvis: Joining adversarial training with vision transformers in next-activity prediction, IEEE Transactions on Services

Computing (2023) 1–14. doi:10.1109/TSC.2023.3331020.

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

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

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

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

[18] S. Abnar, 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.

[19] N. Nader, F. E.-Z. EL-Gamal, M. E. l, Enhanced kinship verification analysis based on color and texture handcrafted techniques, Research Square (2022). doi:https://doi.org/10.21203/rs.3.rs-2139523/v1.

[20] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, Using convolutional neural networks for predictive process analytics, in: 1st International Conference on Process Mining, ICPM 2019, IEEE, 2019, pp. 129–136. doi:10.1109/ICPM.2019.00028.

[21] Z. A. Bukhsh, A. Saeed, R. M. Dijkman, Processtransformer: Predictive business process monitoring with transformer network, CoRR abs/2104.00721 (2021).