electronics

MDPI

Article

Convolutional Neural Networks in Process Mining and Data Analytics for Prediction Accuracy

Ekene Obodoekwe [1], Xianwen Fang [1,2,*] and Ke Lu [1]

[1] School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan 232000, China; eobodoekwe@yahoo.com (E.O.); 2019100059@aust.edu.cn (K.L.)
[2] Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Huainan 232000, China
* Correspondence: xwfang@aust.edu.cn

Abstract: For the reliable prediction and analysis of large amounts of data, big data analytics may be applied in many disciplines. They facilitate the discovery of important information in large amounts of data that would otherwise be obscured. Almost all organizations stored their data in the cloud as event logs over the last few decades. These data can be utilized to extract useful information, which can be used to boost an organization's productivity and effectiveness by identifying, monitoring, and optimizing its processes. Supporting operations, recognizing faults in running processes, predicting event length, and predicting the next activity are all ways of accomplishing this. As part of our strategy, we provide a data collection and machine learning technique. Process mining can help you achieve these objectives. The major enabler of data-driven approaches in process mining is predictive process monitoring. Deep learning has been used in the realm of predictive monitoring to provide accurate future activity predictions in a running trace by analyzing data from previous events. Using image-based data engineering and convolutional neural networks, the next activity in a business process has been forecast in this paper (CNN). The use of CNN in process mining and data analytics guarantees that the proposed system has high accuracy in predicting the next activity in a business process. The experimental evaluation shows that the proposed CNN algorithm is faster at training and inference than the Long Short-Term Memory (LSTM) approach, even when the process has longer traces.

Keywords: CNN; predictive process analytics; next activity prediction; spatial data; business process

Citation: Obodoekwe, E.; Fang, X.; Lu, K. Convolutional Neural Networks in Process Mining and Data Analytics for Prediction Accuracy. *Electronics* **2022**, *11*, 2128. https://doi.org/10.3390/electronics11142128

Academic Editors: Taiyong Li, Wu Deng and Jiang Wu

Received: 14 June 2022
Accepted: 5 July 2022
Published: 7 July 2022
Retracted: 24 July 2023

1. Introduction

Process mining is used to compare the events of a processor to enhance the process. In process mining, event logs are collected, which contain a set of events, including the activity, time-stamp, and case identifier, as well as case attributes, if available. These data should be from the same case, or the event attributes should be similar for all the events. Data analytics is used to analyze, clean, transform, and model the data to discover some useful information that can be used to reach a conclusion and to support perfect decision-making, which helps to effectively maintain the business processes. Advancements in information systems enable the management of an enormous number of event logs for a business process. The data extracted from the event logs are analyzed by the process-mining approach and provide a better understanding of the processes to the business developers and end-users. Recently, in process mining, predictive process monitoring has been considered the main enabler of data-driven approaches. The prediction of future events in a business is extremely important to facilitate seamless decision-making in varying environments and reduce the effect of uncertainty. In business process analysis, predictive analytics is applied to predict running traces using the patterns associated with historical event logs. This can be carried out by predicting the next activity in an activity domain,

This paper has been retracted. A retraction notice was published on 24 July 2023 in *Electronics* **2023**, *12*, 3197. https://doi.org/10.3390/electronics12143197

its timestamp, and the time needed for completion of a cycle until a trace is determined. The major advantage of performing this task is that, by predicting the next activity, the expectations of the next activity are guaranteed to be achieved and, by predicting the remaining cycle time, one can avoid violating deadlines.

There are some existing solutions for predictive process monitoring based on process discovery algorithms using formal language models such as transition systems and Petri nets. These methods demonstrate the way of executing the logged processes. The drawback of such methods is that the situations of pre- and post-activities are hard to describe since the real business processes are unstructured [1]. The process discovery algorithms mostly produce spaghetti-like models [2,3] that are difficult to predict. Moreover, the growth of machine learning in predictive analytics induces everyone to use it to predict business process activities by analyzing the history of events to obtain accurate perceptions of future activities. Machine learning approaches such as the Naive Bayes classifier [4], predictive clustering tree inducer [5], and parametric regression [6] are already being explored in predictive process analytics. These techniques make use of the patterns obtained from the activity sequences of a running trace, the time taken for its execution, and other traces of data from business processes that are accessible at the time of execution.

In addition to machine learning approaches, deep learning approaches such as Deep Neural Networks have recently attracted attention in the predictive analytics of process mining [7]. Deep Learning is a category of neural network architecture in which the input is given as metadata, which is then processed using several layers to obtain the outcome. A major advantage of using deep learning is the presence of a unique function called automatic feature extraction, which can be applied to solve a variety of complex problems. In this paper, business process behavior is predicted using a Deep Learning-based Convolutional Neural Network by developing a predictive process monitoring model.

The predictive process monitoring of a business process is conducted by predicting the next activity in the running cases of a business event based on the event logs and the process execution data. If the final state of a running case is predicted in advance, then the business manager can create valuable outcomes by avoiding any unwanted delays or barriers in the process. At present, this prediction trending due to its benefits for business management and the availability of many previous process execution data [8]. Figure 1 shows the method of process monitoring and control.

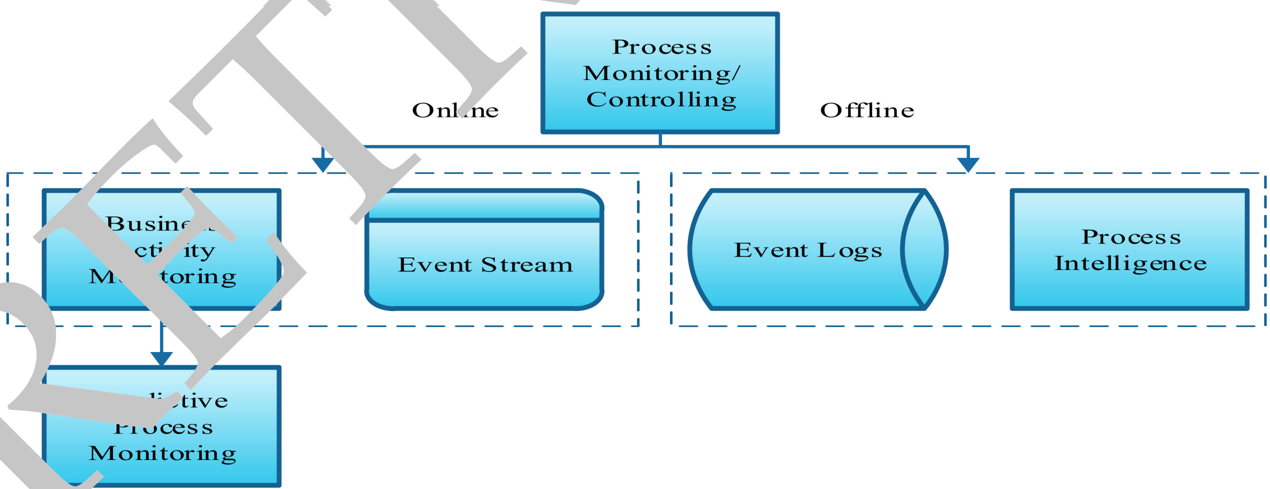


Figure 1. Process monitoring and control.

In this paper, a convolutional neural network algorithm is used to predict the next activity in an event trace from the event logs of a business process. To achieve this, a data engineering scheme is first designed to detect the spatial structure in the sequential order of event traces and transform these into spatial data. In other words, each trace of the history of event logs is converted into a set of prefix traces, which are then transformed into 2D images. The 2D images generated for the prefix traces of a historical event log are trained using the CNN architecture to generate a deep learning model to predict the next activity in an ongoing process. The feasibility of this predictive analysis was investigated using Helpdesk event logs and illustrates that this CNN-based predictive analytic model provides highly accurate next-activity prediction compared with the existing methods briefed in the literature and obtained with the deep learning-based LSTM approach. The remaining part of this study is organized as related works that describe the literature review of various existing methods that are carried out in the predictive process monitoring task in Section 2, a proposed methodology that describes the data engineering scheme and the involvement of CNN architecture in the prediction of the next activity in Section 3, and, finally, this study is concluded in Section 4.

2. Related Works

A framework for the prediction of business process monitoring is described in [9]. This provides the users with continuous recommendations and predictions related to the business activities that are to be performed and the input values needed, and minimizes the possibility of violating business constraints. The constraints of the business process can be specified using Linear Temporal Logic (LTL), even during the process execution. This framework depends on the sequence of the activities performed in an event and the attributes obtained after the activity. The method can be applied to both recommendation and prediction. A decision tree algorithm is used for this purpose. To predict an activity, the decision tree evaluates the likelihood of its satisfying the business constraints for the given attributes. To recommend an activity, the decision tree algorithm selects the attributes that maximize the likelihood of satisfying business constraints. This framework is implemented in the ProM toolset and validated using a real-life dataset of cancer patients, obtained from a Dutch academic hospital. The use of Long Short-Term Memory (LSTM) neural networks in predictive process monitoring by considering the process metrics is explained in [10]. The next activity in a process and its time-stamp are predicted in this paper. Additionally, this method predicts the case results and aggregate performance indicators. In this paper, the author suggests that the use of deep learning approaches is the next step in the research into predictions of the next activity and the remaining time in business processes. In [11], a framework is established that supports the prediction of business processes. The next activity is predicted by training the model with historical logs containing previous processes and this method is designed so that the results can be visualized. The domain experts compare these visualizations based on their experience. The model analysis technique is used for complex visualizations.

A weaker bias-based predictive modeling system named RegPFA artifact is designed and described in [12]. It has two components, namely, the RegPFA Predictor, which acts as the predictive model, and the RegPFA Analyzer, which performs visualization and analysis. The probabilistic model is fitted into a dataset that holds details of previous activity, which help to predict the future of the currently running activity. A visualization of the model is also designed to verify how the proposed probabilistic model works. This model was evaluated on both synthetic and real-world datasets and was found to be effective and outperform suitable benchmarks. Ref. [7] uses a deep recurrent neural network (RNN) to predict the next activity and case remainders in the business process. The specialization of this approach is that it uses separate data for training and validation to eliminate the over-fitting problem, empirically assessing the parameters of neural networks, understanding and visualizing the states of neural networks, and encoding the information regarding timings. Most of the existing methods for the prediction of the next activity use logs of event behavior that have been completely executed; however, in [8], the next activity prediction is based on an analysis of the running events that have not yet been completed. This framework is designed for transforming event forecasting using the sliding window method. Process mining notifies the future activities of a running event. The ability to foresee future insights helps with decision-making.

The completion time of an ongoing process was predicted using process mining, which is demonstrated in [6]. This approach uses a configurable set of abstractions that help balance the problem of under-fitting and over-fitting. The model was trained on real-life event logs and implemented in ProM. The developed model was evaluated using real-life cases from two Dutch municipalities. This method differs from the existing approaches as the predictive model is explicit, can adjust the degree of abstraction based on the requirements and amount of data, and provides improved diagnostics and a better performance. The cycle time prediction was performed in [6] to answer the question, "When will the case be finished?" This method uses a parametric regression of the data available in the logs. By comparing the current event with the log of past events, the remaining cycle time can be predicted. This regression-based system was found to be better than the Naive approach in terms of performance but needs improvements when no case-data variables are assumed. The assumption of the case-data variable is a human-based approach since it requires more knowledge of the process. In [13], the remaining time of a service execution was carried out using stochastic Petri nets with arbitrary firing delays. This method considers the previous event data along with the expected event information. This makes the system predict the remaining time with better quality. Implementation was conducted using the ProM tool. As this method employs Petri Nets, the parallelism in the business process can be naturally captured, as can the resources.

In [2], the process mining approach is shown to be applicable to both structured and unstructured processes to discover and improve the processes. An example of the structured process is the Lasagna process, and an example of the unstructured process is the Spaghetti process, which has been explained in the paper. The process discovered in the Lasagna process is the initiation point for a wide range of analysis techniques used to improve the process. The spaghetti process analysis is challenging, although the probable benefits are significant. The distributed learning of process models for prediction and recommendation of the next activity through "Nested Prediction Model" learning, based on the Nave Bayes classifier, are given in [4]. In this method, the frequent and recurrent activity sequences are first identified and, for each sequence, the predictive model is learned. By using a parallel and distributed solution, huge event logs are processed, which enable realistic decision-making without a perfect model. The datasets used in this system are the BPI2013, which holds the event logs of Volvo IT Belgium, and the BPI2015, which holds the event logs of five Dutch municipalities. In [3], a co-training technique for multiple view methods based on process mining was presented. Here, the author shows that there are many process mining algorithms used to mine the event logs and deliver valuable models to describe the process execution. The developed models are similar to the spaghetti process and are difficult to recognize, as complex, real-life events are more flexible and less structured based on the expectations of stakeholders. This type of model is only used when all probable actions are combined into a single model, resulting in a set of traces being immediately considered in the event log. This issue can be eliminated by the use of trace clustering in preprocessing. Trace clustering means that the event log is split up into similar traces of a cluster to handle various actions and supports, discovering the process models. The developed clustering model is evaluated using machine learning and process mining metrics.

The authors of [14] declare that several techniques use distinct datasets, experimental setup, evaluation metrics, etc., to overcome the problems with the monitoring of predictive processes, but they all result in a poor capability and an uncertain depiction of the advantages and applicability of various methods. Hence, to solve this issue, a detailed survey of outcome-oriented predictive process-monitoring techniques and their evaluation procedures are described in this paper. The review results are more reliable and accurate regarding the Area Under the ROC Curve (AUC) while using lossy sequence encoding. A deep-learning-based prediction of the next event using a gated convolutional neural network (GCNN) and a key-value-predict attention network (KVP) is described in [15]. This method makes use of process data properties such as repetitiveness, variation, and sparsity for evaluation and describes the impact of these properties on the prediction accuracy of various deep learning structures. This method is evaluated for classification properties such as the generalization and handling of class imbalances. This paper guides the researchers in selecting, validating, and benchmarking the models for predictive process monitoring. This paper also highlights the need for various process data properties in the history of events and the reporting of various performance indicators to attain the desired outcomes. In this paper, the research is continued using convolutional neural networks, which are a robust class of deep neural network architecture that can be used for several applications related to speech recognition and computer vision [16,17]. Some researchers proved that CNN provides a higher accuracy for image data with a clear spatial structure [18,19]. The CNN algorithm works well due to the local filtering and max-pooling layers in its architecture.

A convolutional neural network (CNN) is a deep learning neural network designed to process structured arrays of data, such as images [20]. Convolutional neural networks are especially adept at detecting image patterns, such as lines, gradients, circles, and even eyes and faces. Deep learning has been implemented in the field of predictive monitoring to produce accurate predictions of future activity in running trace by studying data from past events. Using CNN in process mining and data analytics ensures the high accuracy of the proposed system in predicting the next business process activity. Even with lengthier traces, the experimental evaluation demonstrates that the proposed CNN process is faster at training and inference than the Long Short-Term Memory (LSTM) method. The classifier utilized CNN to retrieve the data's high-level attributes. The proposed model comprises a series of fully integrated CNNs and layers.

3. Proposed Methodology

The methodology of the proposed business process predictive model is explained as follows: In this method, an image-like data engineering scheme is first structured, which converts the set of prefix traces into 2D image-like data structures, and then the CNN architecture is applied to the historical log of events to predict the next traces of an ongoing process. An event log is supposed to contain data related to the activities involved in an event of a business process and its duration for completion. An event is characterized by two features: the action performed in an event and its time-stamp, which includes the date and time at which it happens. The activity domain is a set of several different activities that occur in an event, based on which the business process predictive model is constructed. An event log consists of a set of events. Each of these events is associated with a specific, unique tracing. This is also represented as a bag of traces. A trace denotes the business process of a business process execution. The sub-sequence of a trace is called a "prefix trace" which considers the initial state of a trace until its end.

The prefix trace can be depicted from an activity perspective, i.e., the frequency or control flow of an activity and its performance, i.e., time consumption. This paper considers both perspectives to generate 2D image-like structures. The log of events is first converted to a labeled imagery dataset. For each prefix trace of a trace in an event log, a 2D image is constructed to depict the labeled prefix trace with its future activity. The activity channel measures the number of times that an activity takes place in a prefix trace, from the beginning to the end of the activity. The performance channel measures the duration of the activity from the beginning, to its last occurrence in a prefix trace before its end. As this paper mainly focuses on recent activities, the last occurrence of activity is considered in this paper. From this, it is possible to figure out how long the current execution has been occurring.

Consider an event as E, event log as L, activity as A, activity domain as AD, timestamp as Ts, and the trace as T.

Here, an event E_i corresponds to the activity A_i with a timestamp Ts_i.

The trace T is a finite sequence of l distinct events with $Ts_i < Ts_j$, $1 \leq i < j \leq l$.

Let $l = |T|$ events, the prefix trace PT_k is the sequence of first k successive events of T with $1 \leq k < l$.

From a prefix trace with l length, $l-1$ prefix traces could be extracted. Let us now consider A_{k+1} as the next activity of a task related to PT_k. Where $PT_k = E_1, E_2, \ldots, E_k$.

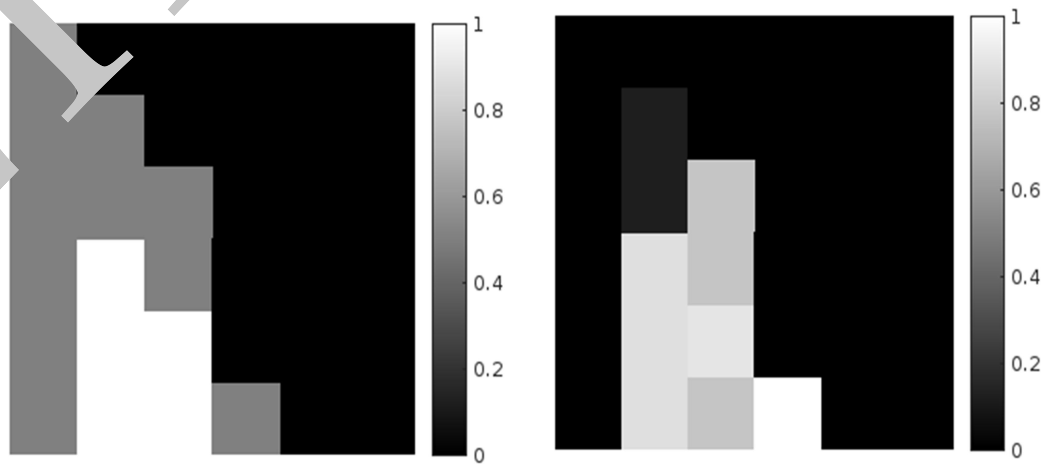
Each event consists of (A_i, Ts_i) pair with $1 \le i \le k$.

PT_k is depicted as a 2D image I_k of $k \times m$, where k is prefix trace length and m is activity domain size. The imagery rows of I_k are successive indices of events, while the imagery columns of I_k are the various activities of an activation domain. Every imagery pixel (x, y) of I_k is a 2D vector consisting of an activity channel as its row and a performance channel as its column. The activity channel is the measure of the number of times an activity occurred in PT_k from Ts_1 to Ts_x, whereas the performance channel is the measure of the time duration between Ts_1 and Ts_x, until the last activity before PT_k. The reason behind the contemplation of the latest occurrence of inactivity is that, in this paper, the recent predictions are used to predict the next activity. Using this method, the recent activities are predicted in the proposed method.

An event log fragment is given in Table 1. This provides the trace ID, activity domain, and timestamp of a helpdesk. Six activities are considered in the experiment. These are raise ticket (R), inspect ticket (I), verify ticket (V), decision (D), reject ticket (RJ), and accept ticket (AT). Each of these events is linked to a particular trace, which resembles an activity in the activity domain and its equivalent timestamp. Table 2 provides a data matrix of the prefix traces of activity and performance channels and imagery representations of both the activity channel and performance channels are given in gray scale in Figure 2.

Table 1. Event log fragment with prefix trace and next activity.

Trace ID	Prefix Trace ID	Prefix Trace	Next Activity
1	1	(R,2021-10-15:09:15)	V
1	2	(R,2021-10-15:09:15), (V,2021-10-16:10:16)	I
1	3	(R,2021-10-15:09:15), (V,2021-10-16:10:16), (I,2021-10-20:05:05)	V
1	4	(R,2021-10-15:09:15), (V,2021-10-16:10:16), (I,2021-10-20:05:05), (V,2021-10-21:10:20)	I
1	5	(R,2021-10-15:09:15), (V,2021-10-16:10:16), (I,2021-10-20:05:05), (V,2021-10-21:10:20), (I,2021-10-25:18:22)	D
1	6	(R,2021-10-15:09:15), (V,2021-10-16:10:16), (I,2021-10-20:05:05), (V,2021-10-21:10:20), (I,2021-10-25:18:22), (D,2021-10-27:16:11)	RJ



(**a**) Activity Channel (**b**) Performance Channel

Figure 2. 2D Image representation of a prefix trace.

Table 2. Data matrix of prefix trace.

(a) Activity Channel Matrix						
Time/Activity	**R**	**V**	**I**	**D**	**RJ**	**AT**
1	1	0	0	0	0	0
2	1	1	0	0	0	[illegible]
3	1	1	1	0	0	0
4	1	2	1	0	[illegible]	0
5	1	2	2	0	0	0
6	1	2	2	1	0	0
(b) Performance Channel Matrix						
Time/Activity	**R**	**V**	**I**	**D**	**RJ**	**AT**
1	0	0	0	0	0	0
2	0	0.9611	0	0	[illegible]	0
3	0	0.9611	6.1736	0	0	0
4	0	7.0002	6.1736	[illegible]	0	0
5	0	7.0020	7.1736	0	0	0
6	0	7.000[illegible]	6.1732	8.0111	0	0

The generated imagery dataset is now trained with a CNN architecture as a business process analytic, customized to predict the next activity in an ongoing trace. There is a problem with the use of deep learning analytics in the predictive process: the training dataset should have images of the same size, whereas the number of rows in an image of a prefix trace can differ according to its length. This problem can be overcome by generating the images with several fixed rows, which are then projected as the length of the longest prefix trace in event history. The empty values are assigned as '0' in these imagery rows. The CNN extends a basic, fully connected, feed-forward neural network model with additional features, such as a convolution layer, pooling layer, and weight sharing. The CNN comprises single or multiple pairs of convolution layers and max-pool layers. The convolution layer in a CNN architecture is placed onto a set of filters, which are simulated over the entire input to process trivial input parts. The output of the pooling layer is a low-resolution form of the output obtained from the convolution layer. In higher layers, several broad filters are used to process the complex regions of a low-resolution input. Finally, the fully connected layer combines all the inputs and produces the outcome.

The generated imagery dataset is now trained with a CNN architecture as a business process analytic, customized to predict the next activity in an ongoing trace, as depicted in Figure 2 (a 2D image representation of a prefix trace). There is a problem with the use of deep learning analytics in the predictive process. The training dataset should have images of the same size, whereas the number of rows in the image of a prefix trace can differ according to its length. This problem can be overcome by generating images with several fixed rows, which are then projected as the length of the longest prefix trace in event history. The empty values are assigned as '0' in these imagery rows. The CNN extends a basic, fully connected, feed-forward neural network model with additional features, such as a convolution layer, pooling layer, and weight sharing. The CNN comprises single or multiple pairs of convolution layers and a max-pool layer. The convolution layer in CNN architecture is placed on a set of filters, which are simulated over the entire input to process trivial input parts. The output of the pooling layer is a low-resolution form of the output obtained from the convolution layer. This enables the translation invariance and tolerance to small variations in the pattern positions in the input. In higher layers, several broad filters are used to process the complex regions of a low-resolution input. Finally, the fully connected layer combines all the inputs and produces the outcome, as shown in the Figure 2 (a 2D Image representation of a prefix trace in (a) the Performance Channel). The input of the CNN architecture is fed into the first convolutional layer, where a set of filters are applied. Each filter is a square matrix that serves as the kernel, which uses only a small part of the input, called a node, from the whole image to exploit the hidden local correlation. Feature maps are generated by applying the convolution operator to the input image. The activation function is generated by applying a nonlinear function to each of the feature maps. The values of the feature maps for the first convolution layer are computed by convoluting the input map with the respective kernel and the linear activation function.

After passing an image through a convolutional layer, the output is passed to an activation function. The sigmoid function is a typical activation function, as expressed in Equation (1)

$$h_j^{(1)}(x,y) = f\left(\sum_{(u,v)\in U} w_{j0}^{(1)} h^{(0)}(x+u, y+v) + b_j^{(1)}\right) \quad With\ U = \left\{(u,v) \in \mathbb{N}^2 \middle| 0 \leq u \leq s, 0 \leq v \leq s\right\} \tag{1}$$

where
$h_j^{(1)}(x,y)$ = *Feature Maps*
$h_j^{(0)}(x,y)$ = *Input Map*
$w_{j0}^{(1)}$ = *Respective Kernel*
$I(x,y)$ = *Input Image*
f = *Nonlinear Activation Function*
j = *Node*
s = *Matrix size*
w_{j0} = *Weight of the ma*

The activation function can be any of the nonlinear functions that are differentiable and continuous, which is similar to the back-propagation learning algorithm. The activation function used in this paper is Relu, and is expressed in Equation (2). The ReLU function, also known as the rectified linear unit, is the same as taking the positive component of the input:

$$f(x) = max(o,x) \tag{2}$$

A pooling layer is introduced before each of the convolution layers to attain spatial invariance and minimizes the dimensions of feature maps, preserves relevant details, and removes unwanted information. Usually, a pooling operation would be a summation, averaging, maximum, or combining of such operations. The pooling operation used in this paper is max-pooling, since it provided the best results in some existing studies.

In the proposed approach, three pairs of convolution layer and max-pool layer are used, as shown in Figure 3. The layers at the end are the fully connected structures, which are similar to those of a feed-forward neural network. This layer combines the various local structures extracted in the low layers to generate the final prediction output. The activation function provides the convolutional neural network nonlinearity. In the absence of the activation function, all neural network layers could be reduced to single matrix multiplication. This paper uses a softmax activation function in the output layer, which is expressed as follows:

$$O_j = f(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{H} e^{x_k}} \tag{3}$$

where H = Number of nodes.

When a ReLU function is applied to the output of the first layer, the result is a higher contrast that brings out the vertical lines and gets rid of the noise caused by other features that are not vertical.

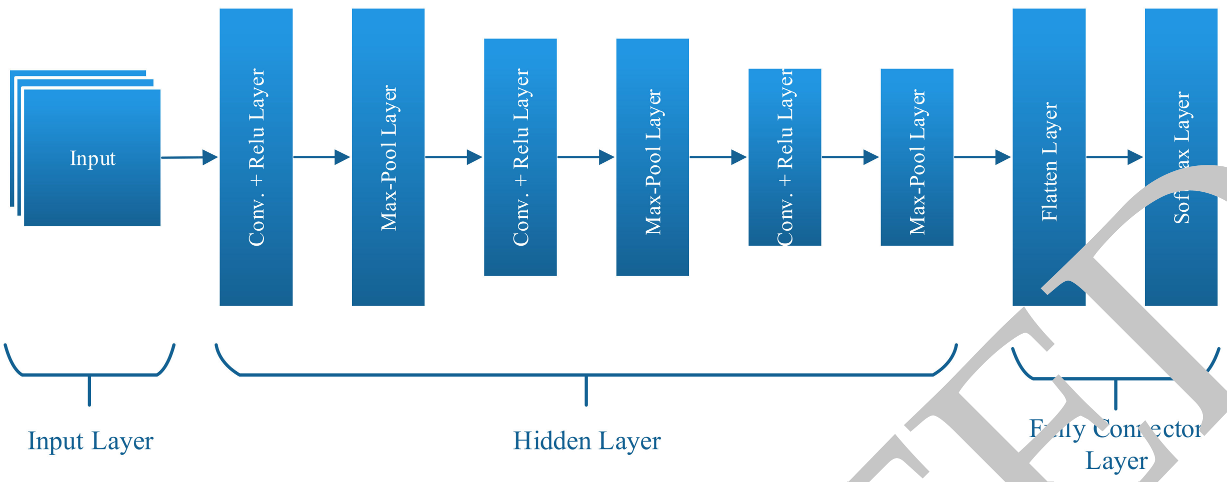


Figure 3. Architecture of convolutional neural network.

4. Experimental Results

The proposed predictive model is implemented in Python and evaluated to predict the next activity. This evaluation aims to first determine the feasibility of transforming spatial data from temporal data, then validate the efficiency of the image-like structures and the prediction accuracy of convolutional neural networks. In this paper, a benchmark dataset named "Helpdesk event log of an Italian Software Company" was used to evaluate the proposed model. This dataset consists of nine activities of a business process, with a total of 13,710 events and 3804 traces of length, ranging from 1 to 14. A fake event is added at the end of each trace. In a ticketing management system, every trace is started with a new ticket. The loss function is minimized by performing Adam Optimization to achieve effective stochastic optimization through the computation of first-order gradients, only with less memory, along with the computation of network weights with batch size 128 based on the running average of the first moment and second-moment estimates. This process is described in Algorithm 1.

Three pairs of convolution layers and a max-pool layer comprise the CNN architecture. The pooling layer is formed of two strides and the sliding window is of size 2 × 2. The generated feature maps are flattened and given to the denser output layer of the softmax unit. The CNN architecture is trained on the same Helpdesk event log benchmark dataset. Batch normalization is performed by transferring the input from the convolution later to the max-pool layer. The application of batch normalization increases the speed of the training process by reducing the sensitivity of the weight initialization without any increase in the overfitting of the training dataset.

After all of the logs are resolved and closed, the traces are removed. A total of 70% of the traces in the event log are taken into consideration for training, while the remaining 30% of the traces are taken into consideration for testing. The following activity may be predicted for each of the testing traces in Figure 4, which depicts the distribution of training and testing classes in the Helpdesk Event Log. Temporal order is maintained between the training trace and the testing trace. This is so that the predictive model can be trained using the past data and its performance can then be evaluated using the incoming data. The distribution of training courses to testing classes can be seen in Figure 4, which is part of the Helpdesk event log. The third activity takes place at the beginning of the traces. In both the training and the testing phases, no labeled prefix traces are present. It was predicted that, after the conclusion of the running trace, this would be the subsequent activity.

Algorithm 1: CNN-based next activity prediction

Require;
1. **Input:** Helpdesk Event log–Dataset
   data of training (that is the log cases;
   dict is unique [i] for I range;
   dataframe represent the concept case, concept, timestamp
   time is the number of iteration; test time,
   Test testing data, maximum trace, activities.
   Train_img: the train model n;
   accuracy is model Evaluation result
2. **Initialize the algorithm**
3. **Import Dataset**
4. **Convert Dataset to Dataframes,**
   unique_data = dataframe[concept].unique()
   dict = {unique[i] for i in range (0, len(unique))}
   for k in dict:
   dict[k]+=1
   dataframe = [case, concept, timestamp]
5. return dataframe **To generate image,**
   train = (training_data, training_time, maximum_trace, activities)
   test = (testing_data, test_time, maximum_trace, activities)
   train_img = array (train)
   test_img = array(test)
   return (train_img, test_img)
6. **To generate label,**
   label_train = getLabel(train)
   label_test = getLabel(test)
   pp = preprocessing.labelEncoder()
   label_train = pp.fitTransform(label_train)
   label_test = pp.transform(label_test)
   return (label_train, label_test)
7. **Convolutional Neural Network**
   model = sequential
   r[illegible] = 0.00[illegible]
   inputShape = (maximum_trace, activities, 2)
   if int(num_of_layers) == i, where i = number of epochs
   model.add(Conv2D(32, (2, 2), inputShape = inputShape, padding=same, kernelInitializer = glorot_uniform, kernelRegularizer = tensorflow.keras.regularizers.l2(reg)))
   model.add(BatchNormalization())
   model.add (Activation(Relu))
   model.add(Max_pool2D(size = (2, 2)))
   (model, reg) ← (generate feature of train and verify input shape)
   end if
8. **Continue the process for 5 epochs with pooling layer of stride 2 and sliding window of size 2 × 2**
   model.add(Flatten())
   model.add(Dense(numOfClasses, activation = softmax, name = actOutput))
   accuracy = modelEvauluation
9. Output: Average of the accuracies of epochs

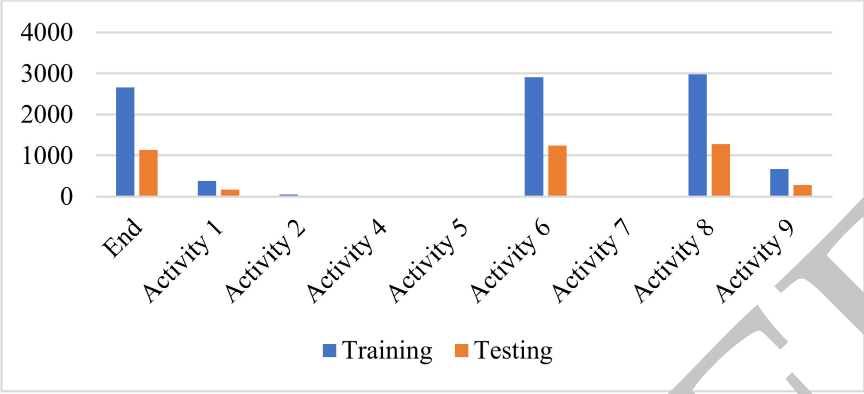


Figure 4. Distribution of training and testing classes in the helpdesk event log.

Overfitting of the training data can be avoided by validating the proposed model with the validation dataset, which is 20% of the training dataset. Once the loss function stops improving the validation dataset for a specific number of iterations, the training procedure will be stopped. Training is performed on five trials, and the performance of the predictive model is calculated by averaging those trials.

The CNN and LSTM deep learning models were implemented on the benchmark Helpdesk event log dataset. We created two files for this purpose, one for training and one for testing the data. In the first section, it was necessary to create an LSTM model to demonstrate prediction using the same dataset. Since we initially created two files, one for training and one for testing, in the initial step, training data were created by designing, training and storing the model. After showing the required information on the screen, we created another routine, which was implemented in the file, loaded the model, and generated random sequences of numbers that were written and evaluated on the screen. Using the same routine as the previous implementation, the user should be allowed to enter a string indicating a prediction operation and, once the string has been appropriately entered into the model, the outcome should be displayed on the screen. In the first phase, we were required to train the model, save it, and display relevant results for the development model process on the screen, as well as from each category, based on the results. The second phase required a model comparison.

Finally, in Table 1, we presented a comparison of the accuracy of the CNN and LSTM models, based on their predictions. Table 3 compares the accuracy of the deep-learning-based CNN model and the LSTM model regarding the benchmark Helpdesk event log dataset. With a 73.93 percent accuracy rate, the proposed CNN predictive model based on deep learning performed better than the LSTM model (22).

Table 3. Performance analysis.

Model	Accuracy
LSTM model	71.23%
Proposed CNN model	73.93%

5. Conclusions

Process mining is a technique for comparing a processor's events to improve the process. The event logs, which include a collection of events, comprising the activity, time-stamp, and case identifier, as well as case characteristics if available, are gathered in process mining. These data should come from the same case, or the event properties should be consistent across all events. Data analytics analyzes, cleans, transforms and models the data to uncover important information, which can be utilized to reach a conclusion and enable excellent decision-making, which aids in the effective operation of corporate processes. This paper describes a Convolutional Neural Network-based next-activity prediction of an event in a business process using process mining and data analytics. Initially, each trace of the historical events is converted into a set of prefix traces, which are then mapped into two-dimensional images. These are called "spatial data". The process data engineering approach is used to convert the temporal data for an event into spatial data, treating them as an image. These are then trained with the CNN to create a model that can predict the next activity in the running processes of a business process. This study illustrates that generating 2D image structures from the traces of event logs is an effective means of modeling the traces in the perception of activity, as well as the performance. The developed predictive model based on 2D images such as data engineering, and CNN produces high accuracy results for the next-activity prediction of a currently running trace in a business process when compared with the LSTM algorithm.

Author Contributions: Conceptualization, E.O. and K.L.; methodology, E.O., X.F. and K.L.; software, E.O.; validation, E.O., X.F. and K.L.; formal analysis, E.O.; investigation, E.O.; resources, E.O.; data curation, E.O.; writing—original draft preparation, E.O.; writing—review and editing, E.O., X.F. and K.L.; visualization, E.O.; supervision, X.F.; project administration, E.O; funding acquisition, X.F. All authors have read and agreed to the published version of the manuscript.

Funding: Natural Science Foundation of Anhui Province, China (No. 2008085QD178).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Van Dongen, B.F.; Crosby, R.A.; van der Aalst, W.M.P. Cycle time prediction: When will this case finally be finished. In *On the Move to Meaningful Internet Systems: OTM 2008*; Meersman, R., Tari, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 319–336.
2. Van der Aalst, W.M.P. Process mining: Discovering and improving spaghetti and lasagna processes. In Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, 11–15 April 2011; pp. 1–7.
3. Appice, A.; Malerba, D. A co-training strategy for multiple view clustering in process mining. *IEEE Trans. Serv. Comput.* **2016**, *9*, 832–845. [CrossRef]
4. Ceci, M.; Spagnoletta, M.; Lanotte, P.F.; Malerba, D. Distributed learning of process models for next activity prediction. In Proceedings of the 22nd International Database Engineering & Applications Symposium (IDEAS 2018), Villa San Giovanni, Italy, 18–20 June 2018; Desai, V.B.C., Fresca, S., Zumpano, E., Masciari, E., Caroprese, L., Eds.; ACM: New York, NY, USA, 2018; pp. 278–282.
5. Pravilovic, S.; Appice, A.; Malerba, D. Process mining to forecast the future of running cases. In *New Frontiers in Mining Complex Patterns—Second International Workshop*, NFMCP 2013, Revised Selected Papers, Ser. LNCS; Appice, A., Ceci, M., Loglisci, C., Manco, G., Masciari, E., Ras, Z.W., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8399, pp. 67–81.
6. Van der Aalst, W.M.P.; Schonenberg, M.H.; Song, M. Time prediction based on process mining. *Inf. Syst.* **2011**, *36*, 450–475. [CrossRef]
7. Evermann, J.; Rehse, J.-R.; Fettke, P. Predicting process behavior using deep learning. *Decision Support Syst.* **2017**, *100*, 129–140. [CrossRef]
8. Verenich, I.; Dumas, M.; la Rosa, M.; Maggi, F.M.; Di Francesco Marino, C. A general framework for predictive business process monitoring. In Proceedings of the International Conference on Advanced Information Systems Engineering, Ljubljana, Slovenia, 13–17 June 2016; pp. 186–202.
9. Márquez-Chamorro, A.E.; Resinas, M.; Ruiz-Cortes, A. Predictive monitoring of business processes: A survey. *IEEE Trans. Serv. Comput.* **2018**, *11*, 962–977. [CrossRef]
10. Hnin, T.; Oo, K.K. Deep Learning for Predictive Process Behavior. In Proceedings of the Sixteenth International Conferences on Computer Applications (ICCA 2018), Beirut, Lebanon, 4–6 October 2018.
11. Becker, J.; Breuker, D.; Delfmann, P.; Matzner, M. Designing and implementing a framework for event-based predictive modeling of business processes. In *Enterprise modelling and information systems architectures-EMISA 2014*; Gesellschaft für Informatik e.V.: Hamburg, Germany, 2014; pp. 71–84.
12. Breuker, D.; Matzner, M.; Delfmann, P.; Becker, J. Comprehensible predictive models for business processes. *MIS Q.* **2016**, *40*, 1009–1034. [CrossRef]
13. Rogge-Solti, A.; Weske, M. Prediction of remaining service execution time using stochastic Petri nets with arbitrary firing delays. In *ICSOC*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 389–403.
14. Teinemaa, I.; Dumas, M.; Rosa, M.L.; Maggi, F.M. Outcome-oriented predictive process monitoring. *ACM Trans. Knowl. Discov. Data* **2019**, *13*, 1–57. [CrossRef]

15. Heinrich, K.; Zschech, P.; Janisch, C.; Bonin, M. Process data properties matter: Introducing gated convolutional neural networks (GCNN) and key-value-predict attention networks (KVP) for next event prediction with deep learning. *Decis. Support Syst.* **2021**, *143*, 113494. [CrossRef]
16. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995; Volume 3361, p. 1995.
17. Prasad, B.; Prasanna, S.M. *Speech, Audio, Image and Biomedical Signal Processing Using Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 83.
18. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation, Inc.: La Jolla, CA, USA, 2012; pp. 1097–1105.
19. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
20. Abdel-Hamid, O.; Mohamed, A.-R.; Jiang, H.; Deng, L.; Penn, G.; Yu, D. Convolutional neural networks for speech recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2014**, *22*, 1533–1545. [CrossRef]