Robust and Generalizable Predictive Models for Business Processes

Praveen Venkateswaran[1(✉)], Vinod Muthusamy[2], Vatche Isahagian[3], and Nalini Venkatasubramanian[1]

[1] University of California Irvine, Irvine, USA
{praveenv,nalini}@uci.edu
[2] IBM Research, Yorktown, USA
vmuthus@us.ibm.com
[3] IBM Research, Cambridge, USA
vatchei@ibm.com

Abstract. Machine Learning models, and more recently Deep Learning models have gained popularity for predictive process monitoring. Predicting the process outcome, remaining time to completion, or the next activity of a running process can be crucial to provide decision information and enable timely intervention by case managers. These models fundamentally assume that the process logs used for training and inference follow the same data distribution and patterns. However, many real-world processes can have gradual or sudden changes, and logs themselves may be associated with different versions of process models modified over time, or customized by different departments with different policies. These can introduce spurious biases and correlations in the data, which can influence predictive models during training and adversely impact their accuracy. In this work, we present RoGen, an approach to train robust predictive models that can identify these spurious correlations and generalize to data with differing distributions. We show that our approach can also be adopted by existing predictive models to improve their robustness and generalizability. We evaluate our approach using real-world event logs and show that even in the presence of spurious data correlations, our models remain robust and outperform existing predictive models.

1 Introduction

There has been an increase in the incorporation of machine learning models in numerous application domains, including business processes. They can be used to predict process outcomes, remaining time to completion or even the next activity of running processes which are important for case managers. The goal of any machine learning model is to learn complex prediction rules using the various features or attributes in a given training dataset for future predictions. This could be either a classification task for discrete predictions, or a regression task for predicting continuous variables. Predictive models learn from features or attributes in the training data that have a significant correlation or causal

A. Polyvyanyy et al. (Eds.): BPM 2021, LNCS 12875, pp. 105–122, 2021.
https://doi.org/10.1007/978-3-030-85469-0_9

relationship with the target variable. Such *invariant* features are those whose correlations with the target are strong in any new test data, thus enabling accurate predictions.

However, there are numerous recent examples highlighting the brittleness of models that are trained using traditional approaches [7,13,20,23,26]. This can be attributed to the fact that real-world data used for training often have inherent data biases due to information or sampling bias, confounding factors, etc. These can introduce *spurious* correlations in features that do not have a causal relationship with the target to be predicted. Moreover, since model training involves minimizing the training error, this leads to models absorbing all correlations (both invariant and spurious) found in the training data. This influence of spurious correlations cause models to perform poorly in test data where these spurious biases no longer hold. Moreover, when these models are trained on data with a specific distribution, and have to generalize to data with slightly different distributions, they can often fail.

A classic example on the need for robust and generalizable ML models was highlighted by [3] where a model, trained to classify images of cows in pastures and camels in the desert, failed when the backgrounds were switched because it was influenced by the spurious correlation of the background (i.e., green pastures with cows and sandy deserts with camels) rather than relying on the invariant features (i.e., the cows and camels themselves).

Even in business processes, real-world process models can have gradual or sudden changes, such as concept drift [5]. In addition, the logs used to train models could be associated with different versions of process models modified over time, or even be customized by different departments with differing policies. These factors, among others, can result in the presence of spurious correlations while training predictive models for business processes and adversely impact their robustness and generalizability, resulting in poor performance.

To illustrate this with an example, Table 1 shows the four most common case variants from sample event logs of the servicing departments of a hypothetical car dealership with two locations A and B. The dealership provides periodic servicing reminders to its customers who either purchased the car from the dealer (in-house), or purchased elsewhere but use the dealer for servicing (external). In order to retain their in-house customers, the dealership also provides many of them special offers in the form of discounts, extended warranties, etc. Moreover, at location A, these offers are also provided to customers who sign up for their loyalty program.

The dealership wants a predictive model that, given customer information, can predict whether or not they should be sent special offers. From the table we can see that the model would require all features, and not just the activity sequence, in order to generate accurate predictions. However, a model trained on a consolidated log from both locations composed primarily of these four case-variants, would erroneously correlate the car brand X with giving special offers, and brand Y with not providing special offers. This is a spurious correlation, as opposed to the invariant correlation of determining special offers based on the customer type and loyalty program information. The influence of this spurious

correlation might lead the model to incorrectly predict that an in-house customer with a brand Y car should not be given a special offer. Similarly, it might also incorrectly predict that an external customer with a brand X car who is not a loyalty member should be given a special offer.

Table 1. Four most common case variants of the event logs of a car dealership service system from two locations: A (top) and B (bottom)

Case: id (Loc. A)	Timestamp	Activity	Case: cust. type	Case: loyalty member	Case: car brand
1	2/1/21 10:05	Obtain car info.	External	True	X
1	2/1/21 10:30	Email reminder	External	True	X
1	2/1/21 10:35	Special offer	External	True	X
2	10/2/21 13:45	Obtain car info.	In-house	False	X
2	10/2/21 14:10	Email reminder	In-house	False	X
2	10/2/21 14:15	Special offer	In-house	False	X
Case: id (Loc. B)	Timestamp	Activity	Case: cust. type	Case: loyalty member	Case: car brand
3	12/2/21 16:00	Obtain car info.	External	True	Y
3	12/2/21 16:30	Email reminder	External	True	Y
4	15/1/21 10:00	Obtain car info.	external	False	Y
4	15/1/21 10:25	Email reminder	External	False	Y

There have been increasing efforts to develop machine learning models that are robust to spurious correlations and which can generalize to Out-Of-Distribution (OOD) datasets [2,4,19]. However, there has not been much existing work in the context of training robust predictive models for business processes.

In this paper, we present our approach, named RoGen, which uses the concept of Invariant Risk Minimization (IRM) [2] to train robust predictive models. IRM has been commonly applied to computer vision tasks, but it has not been used for sequential data like business process event logs. Furthermore, to the best of our knowledge, this is the first paper to develop an approach to train robust and generalizable predictive models for process mining logs that can identify and handle spurious data correlations.

We also show how our approach can work even for training existing predictive models, using a model by [6], and demonstrate the improvement in robustness. We evaluate the performance of the models trained with our approach on real-life event logs against several baselines.

The following section provides background on predictive monitoring, machine learning, and IRM. Section 3 discusses related work in predictive monitoring for business processes. We present our approach in Sect. 4 and evaluate its effectiveness in Sect. 5. Section 6 concludes the paper and discusses opportunities for future work.

2 Background

In this section, we define several elements of process mining and different predictive monitoring tasks. We also provide background on the concept of Invariant Risk Minimization (IRM) that we use to train machine learning models like RNNs and LSTMs that are suited for sequence predictions.

2.1 Event Logs, Traces, and Sequences

Let $\mathcal{A}$ be the set of process activities, $\mathcal{C}$ be the set of case identifiers, $\mathcal{T}$ be the set of timestamps, and $\mathcal{D}_j$ be the set of attributes or features, $1 \leq j \leq m$, where each attribute $d_j \in \mathcal{D}_j$ can be either categorical or numerical. We also define $U = \mathcal{A} \times \mathcal{C} \times \mathcal{T} \times \mathcal{D}_j$ as the event universe.

Definition 1 (Event). *An event $\epsilon_i \in U$ is a tuple $\epsilon_i = (a_i, c_i, t_i, d_{i1}, ..., d_{im})$, where $a_i \in \mathcal{A}$ is the process activity, $c_i \in \mathcal{C}$ is the case identifier, $t_i \in \mathcal{T}$ is its timestamp, and $d_{ij} \in \mathcal{D}_j$, $1 \leq j \leq m$, are the event attributes. Given an event ϵ_i, we define the projection functions $\pi = \{\pi_{\mathcal{A}}, \pi_{\mathcal{C}}, \pi_{\mathcal{T}}, \pi_{\mathcal{D}_j}\}$ where $\pi_{\mathcal{A}} : \epsilon_i \rightarrow a_i$, $\pi_{\mathcal{C}} : \epsilon_i \rightarrow c_i$, $\pi_{\mathcal{T}} : \epsilon_i \rightarrow t_i$, and $\pi_{\mathcal{D}_j} : \epsilon_i \rightarrow d_{ij}$.*

Definition 2 (Trace). *A trace is a non-empty sequence of events $\sigma = \langle \epsilon_1, ..., \epsilon_n \rangle$, $\forall\, \epsilon_i \in U$ and $n = |\sigma|$, such that for all pairs of events ϵ_i, ϵ_j in a given case, where $1 \leq i < j \leq |\sigma|$, $\pi_{\mathcal{T}}(\epsilon_i) \leq \pi_{\mathcal{T}}(\epsilon_j)$ and $\pi_{\mathcal{C}}(\epsilon_i) = \pi_{\mathcal{C}}(\epsilon_j)$.*

Definition 3 (Trace prefix and suffix). *Given a trace $\sigma = \langle \epsilon_1, \epsilon_2, ..., \epsilon_n \rangle$, the prefix of length k is $\sigma_p^k = \langle \epsilon_1, \epsilon_2, ..., \epsilon_k \rangle$, and the suffix of length k is $\sigma_s^k = \langle \epsilon_{k+1}, ..., \epsilon_n \rangle$, where $n = |\sigma|$ and $1 \leq k < n$.*

Definition 4 (Event log). *An event log is a set of traces $\mathcal{L} = \{\sigma_1, ..., \sigma_l\}$ such that each event appears at most once in $\mathcal{L}$.*

2.2 Predictive Monitoring Tasks

Predictive models, given a prefix σ_p^k of a running case, aim to predict a future event ϵ_{k+1}, or a suffix σ_s^k of future events. Existing work has looked at four kinds of predictive monitoring tasks - (1) Next activity prediction, (2) Next timestamp prediction, (3) Activity suffix prediction, and (4) Remaining time prediction. To define these, let Ω be a predictive model, σ_p^k be a trace prefix as defined above, ϵ' be a future predicted event, and $\oplus$ be the concatenation operator between two sequences.

Definition 5 (Next activity). *Given a trace prefix, the model predicts the next activity of the trace, defined as $\Omega_{\mathcal{A}}(\sigma_p^k) = \pi_{\mathcal{A}}(\epsilon'_{k+1})$.*

Definition 6 (Next timestamp). *Given a trace prefix, the model determines the timestamp of the next activity of the trace by predicting its duration as $\Omega_{\mathcal{T}}(\sigma_p^k) = \pi_{\mathcal{T}}(\epsilon'_{k+1}) - \pi_{\mathcal{T}}(\epsilon_k)$.*

Definition 7 (Activity suffix). *The model predicts the activity suffix of a running case by recursively predicting the next activity for multiple future activities. This can be denoted as* $\Omega_{AS} = \langle \Omega_{\mathcal{A}}(\sigma') = \pi_{\mathcal{A}}(\epsilon'_i) | \sigma' = \sigma_p^k \oplus \langle \epsilon'_{k+1}, ..., \epsilon'_{i-1} \rangle \rangle$.

Definition 8 (Remaining time). *The model predicts the remaining time duration of a running case, by recursively predicting the duration of each future predicted activity. Let* θ *be the sequence of predicted future timestamps where* $\theta = \langle \Omega_{\mathcal{T}}(\sigma') = \pi_{\mathcal{T}}(\epsilon'_i) - \pi_{\mathcal{T}}(\epsilon'_{i-1}) | \sigma' = \sigma_p^k \oplus \langle \epsilon'_{k+1}, ..., \epsilon'_{i-1} \rangle \rangle$. *Then the remaining time can be computed as* $\Omega_{RT}(\sigma_p^k) = \sum_{i=k}^{n} \theta_i$.

2.3 Neural Networks and Invariant Risk Minimization

A typical neural network model consists of a layer of inputs $\mathbf{X}$, a layer of outputs $\mathbf{Y}$, and multiple layers in between that are referred to as *hidden* layers. The model optimizes the parameters of its hidden layers θ, while learning a mapping $\mathbf{Y} = f(\mathbf{X}; \theta)$ from the input to output space. In order to train these networks, a loss or risk function $\mathcal{L}(\theta) : \mathbb{R}^n \rightarrow \mathbb{R}$ is used, which maps the model parameters θ to the expected loss on $X \times Y$ for a given function ℓ:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)} \ell(f_\theta(x), y) \tag{1}$$

where $x \in X$, $y \in Y$, and ℓ is a function like cross-entropy loss for classification. The standard Empirical Risk Minimization (ERM) methodology used by existing predictive monitoring approaches tries to minimize the average loss over all training examples. ERM fundamentally assumes that the data is independent and identically distributed (i.i.d) and that the training and test distributions are similar. However, as described in Sect. 1, this may not hold in real-world datasets and it has been shown that in these situations, ERM is not robust and does not achieve good Out-of-Distribution (OOD) generalization. This has motivated the need for alternative approaches to train predictive models, that can identify and handle spurious correlations.

In this paper, to create robust and generalizable predictive models for business processes, we leverage the concept of Invariant Risk Minimization (IRM) [2]. To use IRM, we consider the event logs to consist of $\mathcal{E} = \{e_1, ..., e_n\}$ *environments*, where $1 \leq |\mathcal{E}| < \infty$. Each environment refers to a potential source of spurious correlations, such as logs from multiple departments, different process model versions, etc. We denote X^e, Y^e as the input and output data collected from each environment $e \in \mathcal{E}$. We can then similarly define the loss for each environment $\mathcal{L}_e(\theta)$ as:

$$\mathcal{L}_e(\theta) = \mathbb{E}_{(x^e \in X^e, y^e \in Y^e)} \ell(f_\theta(x^e), y^e), \quad \forall e \in \mathcal{E} \tag{2}$$

Invariant Risk Minimization searches for the invariant set of input attributes across the different environments. As explained in Sect. 1, invariant features have a strong causal relationship with the target variable for any given data, while spurious features may have strong correlations with the target for some data environments, but not in others. Formally, the set of invariant attributes X^I is

defined as one where the target prediction probability is consistent across all environments (i.e.) $p(Y|X_i \in X^I, \mathcal{E})$ is approximately constant. Conversely, the set of spurious attributes X^S consists of features whose prediction probabilities vary across environments due to the presence of spurious correlations. It follows that $X^I \cup X^S = X$, and $X^I \cap X^S = \emptyset$, (i.e.) a feature cannot be both invariant and spurious.

The IRM principle finds an invariant data representation $\Phi : X \rightarrow H$ such that the optimal predictive model with parameters $\theta : H \rightarrow Y$, is the same across all environments $e_i \in \mathcal{E}$ (i.e.) it is not influenced by variations from spurious correlations. Hence, to find a model that optimizes the loss in each environment, while simultaneously identifying invariant feature attributes across environments requires solving the following bi-level optimization problem:

$$\min_{\substack{\Phi: X \rightarrow H \\ \theta: H \rightarrow Y}} \sum_{e \in \mathcal{E}} \mathcal{L}_e(\theta^\top \Phi(X^e)) \tag{3}$$

$$\text{s.t. } \theta \in \underset{\bar{\theta}}{\text{argmin}} \ \mathcal{L}_e(\bar{\theta}^\top \Phi(X^e)), \ \ \forall e \in \mathcal{E} \tag{4}$$

However, since this optimization is highly intractable, particularly when Φ is non-linear, the authors in [2] propose a tractable variant:

$$\min_{\Phi: X \rightarrow Y} \sum_{e \in \mathcal{E}} \underbrace{\mathcal{L}_e(\Phi(X^e))}_{\text{IRM Loss}} + \underbrace{\lambda \|\nabla_\theta \mathcal{L}_e(\theta^\top \Phi(X^e)\|_2^2}_{\text{IRM Penalty}} \tag{5}$$

While training a predictive model using Eq. 5, the IRM Loss term minimizes the training loss in each environment, while the IRM Penalty term balances between the predictive performance of the model within each environment and its invariance across environments using a regularizer $\lambda \in [0, \infty)$. This ensures that a model does not get influenced by spurious correlations which may lead to it performing well for some environments, but not in others. It is trained to balance its performance across environments by identifying the invariant representation, thus leading to robust and generalizable predictive models.

2.4 Sequence Prediction Neural Networks

Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks are popular predictive models for the sequential data in business process event logs since they persist information across sequences unlike traditional neural networks. RNNs have a cyclic structure and can be unfolded as shown in Fig. 1. At each step of the sequence, referred to as timestep t, $\mathbf{x}_t$ is the input and the $\mathbf{h}_t$ is the hidden state which contains information extracted from all the timesteps up to t. RNNs perform well for sequential data by sharing parameters across different parts of the model. In an RNN, the hidden state $\mathbf{h}_t$ is updated using the previous hidden state and the current input at each timestep:

$$\mathbf{h}_t = f(U\mathbf{x}_t + W\mathbf{h}_{t-1} + b)$$

Then the output $\mathbf{o}_t$ at time t is computed as:

$$\mathbf{o}_t = f(V\mathbf{h}_t + c)$$

where f is a non-linear activation function (e.g.) tanh or sigmoid, and U, W, V are the weight parameters and b, c the biases of the new input and hidden state. However, RNNs have been shown to perform poorly for long sequences and retaining long-term dependencies, a phenomenon called *catastrophic forgetting*.

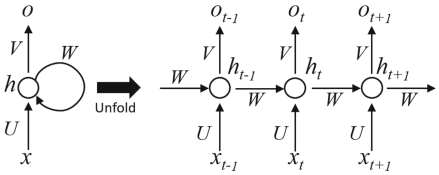

Fig. 1. Unfolding an RNN

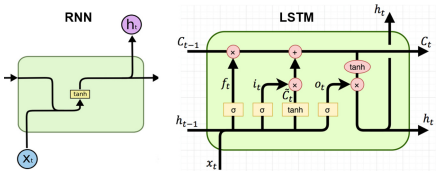

Fig. 2. LSTM vs RNN

Long Short-Term Memory (LSTM) architectures are a special kind of RNN that solve this issue and can learn long-term dependencies. Unlike the single layer of RNNs, LSTMs have four interacting layers as shown in Fig. 2. The LSTM model can be described by the following equations where the $\odot$ operator denotes the Hadamard element-wise product:

$$\begin{aligned}
\mathbf{f}_t &= \text{sigmoid}(W_f\mathbf{x}_t + V_f\mathbf{h}_{t-1} + b_f) \\
\mathbf{i}_t &= \text{sigmoid}(W_i\mathbf{x}_t + V_i\mathbf{h}_{t-1} + b_i) \\
\mathbf{o}_t &= \text{sigmoid}(W_o\mathbf{x}_t + V_o\mathbf{h}_{t-1} + b_o) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tanh(W_c\mathbf{x}_t + V_c\mathbf{h}_{t-1} + b_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{C}_t)
\end{aligned}$$

The LSTM first decides the information to remove from the cell state using the sigmoid layer $\mathbf{f}_t$ also known as the *forget gate*. It uses $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ to output a value between 0 and 1, where 0 denotes completely forgetting information, while 1 denotes completely retaining it. This is followed by the *input gate* $\mathbf{i}_t$ which decides which values to update and a tanh layer generates a vector of new candidate values to be added to the state. The new cell state $\mathbf{C}_t$ is obtained by forgetting some and adding new information. The *output gate* $\mathbf{o}_t$ decides the output of the cell state and also updates the hidden state $\mathbf{h}_t$.

3 Related Work

In this section we first discuss existing work on predictive models for business processes. We then present related work on approaches for generalization of machine learning models used in other domains.

3.1 Predictive Models for Business Processes

The work by Evermann et al. [12] looks at next activity prediction using LSTMs combined with an embedding layer to reduce the dimensionality of the input data and include attributes like the resource associated with each event. Their architecture comprises of the embedding layer with an embedding dimension of 125, followed by two LSTM layers.

Tax et al. [24] use an LSTM based architecture consisting of a shared LSTM layer that feeds two independent LSTM layers, one specialized for predicting the next activity, and the other for predicting the next event timestamp. Their model jointly predicts both the activity and timestamp using a multi-task learning approach, which they show has a better performance than learning each task individually.

Camargo et al. [6] also use an embedding layer similar to [12] along with two LSTM layers. They define the number of embedding dimensions as the fourth root of the number of unique activities. Moreover, like [24] they also use specialized layers for the activity and resource attributes and propose three variants of the baseline architecture which concatenate information at different points in the network and use a similar multi-task learning approach.

Mauro et al. [10] and Pasquadibisceglie et al. [18] use Convolutional Neural Networks (CNNs) for the next activity prediction task. In CNNs, a convolutional layer applies a set of filters that are replicated along the whole input to process small local parts. These filters identify specific patterns or signals and the authors propose schema to represent the running case as two-dimensional images.

Taymouri et al. [25] use Generative Adversarial Networks (GANs) for the next activity and next timestamp prediction tasks. GANs are useful when the amount of available training data is insufficient for effective training of LSTM networks.

3.2 Generalization Approaches

There are various approaches to improving out-of-distribution generalization of deep learning models. Data augmentation techniques aim to make the model more robust by training using instances obtained from neighbouring domains hallucinated from the training domains, and thus make the network ready for these neighbouring domains. Shankar et al. [22] augment data using instances perturbed along directions of domain change and use a second classifier to capture this. Carlucci et al. [7] apply augmentation to images during training by simultaneously solving an auxiliary unsupervised jigsaw puzzle alongside.

Decomposition based approaches represent the parameters of the network as the sum of a common parameter and domain-specific parameters during training. Khosla et al. [14] applied decomposition to domain generalization by retaining only the common parameter for inference. Li et al. [16] extended this work to CNNs where each layer of the network was decomposed into common and specific low-rank components.

Another approach is to pose the generalization problem as a meta-learning task, whereby we update parameters using meta-train loss but simultaneously minimizing meta-test loss. Santoro et al. [21] trained models that adapt using small amounts of labeled data from the new domain, while Dou et al. [11] considered distribution shifts in only test data.

4 Our Approach

4.1 Data Preprocessing

In this section we describe our approach to preprocessing the event log to prepare k-prefixes for the training and test data. For the predictive monitoring tasks described in Sect. 2.2, the model has to learn a function that, given a k-prefix $\sigma_p^k = \langle \epsilon_1, ..., \epsilon_k \rangle$, predicts the next activity a_{k+1} and the next timestamp t_{k+1} in addition to the other attributes d_{jk+1}, $1 \leq j \leq m$. This process is then performed recursively to obtain the activity suffix as well as to predict the future timestamps to compute the remaining time of the case. For the timestamp attribute, we use the relative time between activities, calculated as the time elapsed between the timestamps of the current event and its previous event.

The attributes in the prefixes can be categorical or continuous variables. Continuous attributes are typically normalized between 0 and 1 before they are passed as input to the neural network. There are several approaches in the literature to encode a representation of categorical variables e.g. one-hot encoding, label encoding, using embedding dimensions, etc. Authors in [6,12,17] used label encoding followed by embedding dimensions to reduce the dimensions of the input data, while authors in [9,24] use one-hot encoding. The choice of embedding dimensions can impact accuracy, where a small number may not capture feature relations, and a large number may cause model over-fitting. One-hot encoding for features with many unique values, can result in high dimensional input matrices, which can adversely impact model performance.

Since our focus in this work is on feature identification and distinguishing between invariant and spurious features, we represent the categorical features using label encoding which has been shown to perform well on ordinal data such as those found in business process logs. This also ensures that we can handle multiple attributes without a large increase in model complexity or the number of parameters. We note that our approach can easily integrate other encoding approaches as well.

In order to generate the k-prefixes, we use the popular prefix padding approach also used by [6,9,10,18,24], where every possible set of prefixes σ_p^k is considered, where $1 < k \leq n$. The prefixes are padded with zeroes in case they are shorter than the specified vector length. Depending on the size of the dataset, we either set n to be the length of the longest trace or use the n most recent events. We also maintain a vector of lengths of each case which allows us to stop predictions when the case is finished. Table 2 shows an example of the preprocessed inputs, target features and timestamps for a given 4-prefix input. The ϕ symbol denotes the end of the case.

Table 2. Preprocessing of input k-prefix

Input 4-prefix	Input features	Input timestamp	Target features	Target timestamp
$\langle(a_1, 13/1/2021\ 00{:}15\text{AM}, d_{11}, d_{12}, d_{13}),$	$(1, 1, 1, 1)$	0	$(2, 1, 2, 1)$	1500
$(a_2, 13/1/2021\ 00{:}40\text{AM}, d_{11}, d_{22}, d_{13}),$	$(2, 1, 2, 1)$	1500	$(3, 1, 5, 4)$	2280
$(a_3, 13/1/2021\ 01{:}18\text{AM}, d_{11}, d_{52}, d_{43}),$	$(3, 1, 5, 4)$	2280	$(4, 2, 3, 1)$	2700
$(a_4, 13/1/2021\ 02{:}03\text{AM}, d_{21}, d_{32}, d_{13})\rangle$	$(4, 2, 3, 1)$	2700	ϕ	ϕ

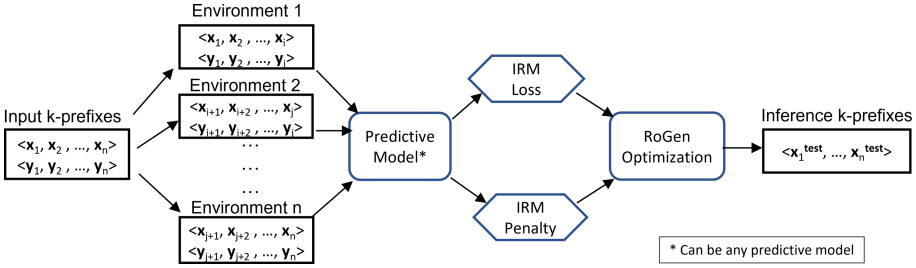

Fig. 3. RoGen training workflow

4.2 RoGen Model Architecture and Training Workflow

In this section, we describe RoGen's model architecture and training workflow. The model architecture consists of an input layer for the k-prefixes from event logs. This is followed by two stacked LSTM layers as described in Sect. 2.4 and a dense output layer. The output layer consists of outputs for predicting both the next activity as well as the next timestamp. RoGen simultaneously optimizes both tasks during training, also known as multi-task or multi-output learning, similar to the approach of [6,24]. This allows RoGen to exploit commonalities and differences across both tasks, which is often present in process trace logs (e.g. activities and their time duration are typically correlated). This multi-task optimization can result in improved training efficiency and prediction accuracy, when compared to training models separately for each task as shown by [6,24].

Figure 3 shows the training workflow of our approach that uses Invariant Risk Minimization (IRM) to train robust and generalizable models as described in Sect. 2.3. The input k-prefixes from the event log are first split into the different environments, which are further divided into training and test environments. The input prefixes and target outputs for the different prediction tasks from each training environment are then passed to the RoGen predictive model. The training algorithm used by the model is detailed in Algorithm 1, where it uses Eq. 5 to compute the training loss (IRM Loss) and penalty (IRM Penalty) for each environment $e \in \mathcal{E}$. The average loss and penalty over all environments are used to optimize the RoGen model using the Adam optimization algorithm. The trained model can then be used for inference on new k-prefixes for robust predictions of future activities and their timestamps. Figure 3 also highlights

that the RoGen model can be easily replaced by any existing predictive model, showing the extensibility of our approach.

Algorithm 1. RoGen Training Algorithm

Require: Distribution over inputs X and targets Y;
Require: s: Total learning steps; f_θ: function to learn
Require: w: Warm up steps; $\mathcal{L}$: Loss function for the prediction error
Require: γ: Learning rate; r: Regularization weight; p: IRM Penalty weight
Require: θ: Model parameters ; μ: Mean function

for $i = 1 \rightarrow s$ **do**
  Sample env e k-prefixes $X^e, Y^e = \langle \mathbf{x}_i, ..., \mathbf{x}_j \rangle, \langle \mathbf{y}_i, ..., \mathbf{y}_j \rangle \ , \ \forall e \in \mathcal{E}$
  $l_e = \mathcal{L}_e(f(X^e), Y^e) \ , \ \forall e \in \mathcal{E}$ ▷ IRM Loss for each environment
  $L2 = \|\theta\|_2$ ▷ L2 regularization
  $l_e^{pen} = \text{IRMPenalty}(X^e, Y^e) \ , \ \forall e \in \mathcal{E}$ ▷ Equation 5
  if $i > w$ **then**
    $l_{final} = \mu(l_e) + rL2 + p(\mu(l_e^{pen})) \ , \forall e \in \mathcal{E}$ ▷ Total loss
  else
    $l_{final} = \mu(l_e) + rL2 \ , \forall e \in \mathcal{E}$
  end if
  $\theta = \theta - \gamma \nabla \theta l_{final}$ ▷ Optimization
end for

5 Evaluation

We implemented RoGen in Python using PyTorch 1.7.0 and evaluated its performance using five real-life event logs. We experiment using two versions of each event log - first with the original data and attributes, and then augmenting it with an additional spurious attribute to test the robustness and generalizability of the approaches. Our code and data are available[1].

We compared RoGen's performance against four baselines [6,10,12,24] based on their publicly available implementation. Furthermore, to highlight the extensibility of our approach and evaluate its performance when applied to an existing predictive model, we also incorporated the model by Camargo et al. [6] into our training workflow in Fig. 3, which we named RoGen-C in the experiments. We modified the baseline approaches to use the same set of log attributes to perform a fair evaluation.

5.1 Experimental Setup

Datasets: We used five publicly available real-life event logs. Table 3 highlights the characteristics of these logs and we describe them below.

[1] https://github.com/praveenv/RoGenBPM.

Table 3. Event logs description

Event Log	Num. traces	Num. events	Num. activities	Avg. activities per trace	Max. activities per trace	Mean duration	Max. duration
Helpdesk	4579	21344	14	4.66	15	40.9 days	59.9 days
BPI 2013	1487	6660	7	4.47	35	178.9 days	2254.8 days
BPI 2015	5647	262628	356	46.50	154	101.4 days	1512.0 days
BPI 2018	43809	2514266	41	57.39	2973	335.3 days	1011.3 days
BPI 2019	251734	1595923	42	6.34	990	71.5 days	25670.5 days

- **Helpdesk:**[2] Contains traces from a ticketing management process of the helpdesk of an Italian software company.
- **BPI 2013:**[3] Contains traces from an incident and problem management system at Volvo IT Belgium.
- **BPI 2015:**[4] Consists of five event-logs containing traces from building permit applications provided by five Dutch municipalities during a period of four years.
- **BPI 2018:**[5] Contains traces of payments for German farmers from the European Agricultural Guarantee Fund over a period of three years. Over the years, there are changes in the process model due to changes in EU regulations. The traces are from four different departments and each of them may have implemented their processes differently.
- **BPI 2019:**[6] Contains traces from an MNC in The Netherlands depicting purchase order handling processes for paints and coatings with different flows in the data. Since the BPI 2018 and BPI 2019 datasets are extremely large, we use a random 10% sampling of data for our experiments.

Spurious Attribute and IRM Environments: To specifically evaluate the robustness of all approaches, we augment every event log with an additional numeric spurious attribute which is a common evaluation methodology [1,8,15]. We also divide the logs into environments for the IRM-based approaches. For the Helpdesk, BPI 2013, and BPI 2019 event logs, we divide them into three environments, two for training with 35% of data each, and the third as test with the remaining 30%. For BPI 2015, we treat logs from each of the five municipalities as an environment, and use four for training and one as test. Similarly, for BPI 2018, logs from the four departments are used as environments, and we use three for training and one as test. This results in BPI 2015 and BPI 2018 having unequal sizes of each environment unlike the other logs, allowing us to demonstrate the effectiveness of RoGen even with unbalanced data distributions.

[2] https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb.
[3] https://doi.org/10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11.
[4] https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1.
[5] https://doi.org/10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972.
[6] https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1.

In each of the above logs, we identify three common case variants (denoted as A, B, C) that have the highest occurrence rates in the event log. For each environment $e \in \mathcal{E}$, we define the spurious correlation between each variant and a specific value V_i of the spurious feature (i.e.) $p(A|V_1, e) = p(B|V_2, e) = p(C|V_3, e) = \alpha_e$, where V_1, V_2, V_3 are values of the spurious attribute and α_e is the strength of the spurious correlation in environment e. For every other case variant, V_i is set to a random number. We set $\alpha_e = 0.9$ as the highest spurious correlation in one of the training environments, and reduce it by 0.05 for every subsequent training environment. We then set $\alpha_e = 0.1$ for the test environment. We note that approaches that do not require explicit environment definitions can also be used similarly [27].

In the training environments, we set a high value of α_e to build a strong spurious correlation between the common case variants and the spurious attribute. However, in the test environment, for the spurious correlation to no longer hold, we set a low value of α_e to mimic a change in data distribution. As described in Sect. 2.3, the varying values of α_e for the spurious attribute in the training environments is detected by IRM to identify the spurious correlation, and ensure that the model does not get influenced by the spurious attribute. Models that incorrectly get influenced by the strong spurious correlation in the training environments, will fail to generalize to the test environment and hence demonstrate low robustness. For training the baseline approaches, the training environments are consolidated into a single input log.

Evaluation Metrics: We use the same evaluation metrics adopted in the baseline comparison approaches [6,10,12,24]. For the next activity prediction task, we use the percentage of correct predictions over the total number of predictions. For the next timestamp prediction, we report the Mean Absolute Error (MAE) which is the average of the absolute value difference between the predicted timestamps and the ground truth timestamps. For the activity suffix prediction, the Damerau-Levenshtein (DL) edit distance metric is commonly used, which measures the edit distance between two given activity traces without penalizing too harshly any transpositions of activities. This value is then normalized by the lengths of the two traces, obtaining a similarity value between 0 and 1. For the remaining time prediction of a case, we use the average of the MAE obtained for all the recursive next timestamp predictions.

5.2 Results

Table 4. Next activity prediction accuracy (%)

Method	Helpdesk		BPI 2013		BPI 2015		BPI 2018		BPI 2019	
	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.
Tax et al. [24]	78.94	65.07	58.99	46.27	38.66	9.85	68.63	35.04	56.75	35.15
Evermann et al. [12]	78.52	61.48	55.67	44.57	38.18	10.29	62.50	34.89	57.43	38.19
Mauro et al. [10]	79.30	61.70	51.29	41.35	35.02	6.43	64.32	34.33	63.43	39.70
Camargo et al. [6]	**80.57**	66.14	**62.23**	51.54	43.90	10.41	73.41	35.65	73.39	41.47
RoGen	80.37	71.06	61.18	**56.75**	47.48	**33.37**	74.20	56.25	**74.07**	**66.43**
RoGen-C	79.78	**72.07**	61.75	56.23	**52.10**	29.84	**76.74**	**57.27**	73.94	63.30

Next Activity and Timestamp Prediction: Table 4 shows the accuracy percentages achieved by all the approaches in predicting the next activity for both the original event logs (Orig.) as well as the logs with the additional spurious feature (Gen.). For the original logs, we see that RoGen and RoGen-C achieve comparable accuracies to [6] and outperform the other baselines across all the event logs. We also observe that for BPI 2015 and BPI 2018, where the logs were collected from multiple sources, treating them as separate environments using IRM results in an increase in accuracy even when the underlying predictive model is the same (RoGen-C and [6]).

For the logs with the spurious feature, our IRM based approaches outperform all the baselines. We observe that the models of the baseline approaches are influenced by the spurious correlations and have a significant drop in accuracy ranging from an average of 19% for the Helpdesk dataset, to an average of 76% for the BPI 2015 dataset. On the other hand, RoGen and RoGen-C are more robust to the spurious correlation and do not have a large drop in accuracy.

Similarly, Table 5 compares the Mean Absolute Error (MAE) values of all the approaches for the next timestamp prediction task. The MAE values are reported in days and lower error values signify better performance. We only use [6,24] as the baselines since [10,12] do not handle this task. We observe that for a majority of both kinds of event logs – original and with the spurious feature, our approaches outperform the baselines. In particular, RoGen-C achieved

Table 5. Next timestamp prediction Mean Absolute Error (MAE)

Method	Helpdesk		BPI 2013		BPI 2015		BPI 2018		BPI 2019	
	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.
Tax et al. [24]	5.18	7.56	**14.11**	15.13	1.96	1.98	6.12	6.82	6.60	12.01
Camargo et al. [6]	4.99	7.35	16.35	14.50	1.92	2.03	5.31	**6.78**	6.36	11.20
RoGen	5.06	7.40	14.74	14.11	**1.74**	1.95	**3.38**	7.51	6.48	10.09
RoGen-C	**4.95**	**7.19**	15.37	**11.87**	1.92	**1.94**	4.90	7.48	**5.88**	**9.63**

consistently lower MAE than the other approaches for both kinds of logs. From Tables 4 and 5, we can see that models using IRM do not degrade in accuracy even if the logs do not have any spurious data correlations. In addition, when spurious correlations exist in event logs, the IRM based models significantly outperform traditional predictive models.

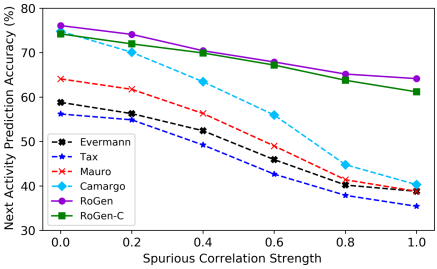

Fig. 4. Comparison of next activity prediction accuracy for varying strengths of spurious correlation

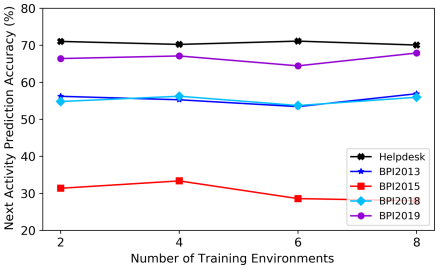

Fig. 5. RoGen next activity prediction accuracy for varying number of environments

Impact of Strength of Spurious Correlations: To understand the impact of the strength of spurious correlations in an event log on model performance, we evaluate the approaches for varying strengths of spurious correlations α_e in each of the two training environments for the BPI 2019 event log. Figure 4 shows the next activity prediction accuracy achieved by all the approaches where the spurious correlation strength refers to the average spurious correlation $\mu(\alpha_e)$ across the two training environments. We see that when there is no spurious correlation (i.e.) $\mu(\alpha_e) = 0.0$, all the approaches have accuracies similar to their performance on the original event log as observed in Table 4. However, when $\mu(\alpha_e)$ is increased, the baseline approaches are influenced by the spurious feature and their test accuracy degrades. We observe for high levels of spurious correlation, the baseline approaches have low accuracy values. On the other hand, RoGen and RoGen-C show good robustness to the increasing levels of spurious correlations and continue to perform well.

Impact of Number of Environments: We evaluate the scalability and robustness of RoGen to multiple sources of spurious correlations, by varying the number of training environments in Fig. 5. For each event log, we use the same strengths of spurious correlation, but vary the number of environments from $\{2, 4, 6, 8\}$. We observe that the accuracy achieved by RoGen does not have much variance even with a larger number of environments. This shows that our approach can handle large and diverse event logs with multiple sources of spurious correlations.

Suffix Prediction: Tables 6 and 7 show the results of the activity suffix prediction and remaining time prediction tasks respectively. RoGen and RoGen-C continue to outperform the baselines, particularly in logs with the spurious feature. For some of the logs, all the approaches achieve similar results since we limited the number of future predictions due to the size of the log.

Table 6. Activity suffix prediction DL similarity

Method	Helpdesk		BPI 2013		BPI 2015		BPI 2018		BPI 2019	
	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.
Tax et al. [24]	0.75	0.68	0.33	0.19	0.14	0.03	0.15	0.06	0.18	0.07
Camargo et al. [6]	**0.76**	**0.72**	0.37	0.27	0.10	0.02	**0.17**	0.06	**0.19**	0.09
RoGen	**0.76**	**0.72**	**0.38**	**0.32**	0.14	**0.09**	**0.17**	**0.13**	**0.19**	**0.16**
RoGen-C	0.75	**0.72**	**0.38**	0.27	**0.15**	**0.09**	**0.17**	**0.13**	**0.19**	0.14

Table 7. Remaining time prediction MAE

Method	Helpdesk		BPI 2013		BPI 2015		BPI 2018		BPI 2019	
	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.	Orig.	Gen.
Tax et al. [24]	17.85	8.84	**22.32**	20.79	61.21	**60.97**	53.46	53.04	38.82	29.92
Camargo et al. [6]	18.35	9.21	23.92	20.94	**60.93**	**60.97**	53.40	52.51	38.53	28.69
RoGen	17.85	8.35	23.44	20.49	**60.93**	**60.97**	**50.74**	52.43	38.55	27.96
RoGen-C	**17.40**	**8.03**	23.09	**20.05**	**60.93**	**60.97**	51.29	**52.00**	**38.48**	**27.76**

6 Conclusion and Future Work

In this paper, we present a novel approach to train predictive models for business processes that are robust and generalizable in the presence of spurious data correlations. Existing work on predictive business process monitoring have not accounted for the presence of spurious correlations in event logs which can arise due to various factors. Since predictive monitoring tasks are often used by case managers, deploying robust models is critical for many real-world business processes.

Our approach uses the concept of Invariant Risk Minimization and we also demonstrate how existing predictive models can utilize IRM to improve their robustness. Our experiments highlight the importance of our approach, where our robust predictive models outperform several existing baselines on real-life logs, especially when they also have varying levels of spurious correlations. We also show that our implementation can easily be used to improve the robustness of any predictive model and our logs with spurious correlations can be used to evaluate robustness.

We intend to extend our work to incorporate and compare other techniques to achieve robustness like meta-learning, data augmentation, adversarial learning, etc. We also plan to improve our approach to handle logs where the sources of spurious correlations may be hard to identify. We also intend to evaluate different kinds of predictive models in this context and also extend our approach to handle other prediction tasks.

References

1. Ahuja, K., Shanmugam, K., Varshney, K., Dhurandhar, A.: In: In: ICML (ed.) Invariant Risk Minimization Games, pp. 145–155. PMLR (2020)
2. Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. Stat **1050**, 27 (2020)
3. Beery, S., Van Horn, G., Perona, P.: Recognition in terra incognita. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 456–473 (2018)
4. Bengio, Y., Deleu, T., Rahaman, N., et al.: A meta-transfer objective for learning to disentangle causal mechanisms. In: ICLR (2019)
5. Bose, R.J.C., Van Der Aalst, W.M., et al.: Dealing with concept drifts in process mining. IEEE Trans. Neural Networks Learn. Syst. **25**(1), (2013)
6. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate lstm models of business processes. In: International Conference on Business Process Management, pp. 286–302. Springer (2019)
7. Carlucci, F.M., et al.: Domain generalization by solving jigsaw puzzles. In: CVPR, pp. 2229–2238 (2019)
8. Choe, Y.J., Ham, J., Park, K.: An empirical study of invariant risk minimization. arXiv preprint arXiv:2004.05007 (2020)
9. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Petrucci, G., Yeshchenko, A.: An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 252–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_15
10. Di Mauro, N., Appice, A., Basile, T.M.A.: Activity prediction of business process instances with inception CNN models. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019. LNCS (LNAI), vol. 11946, pp. 348–361. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35166-3_25
11. Dou, Q., de Castro, D.C., Kamnitsas, K., Glocker, B.: Domain generalization via model-agnostic learning of semantic features. In: Advances in Neural Information Processing Systems, pp. 6450–6461 (2019)
12. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**, 129–140 (2017)
13. de Haan, P., Jayaraman, D., Levine, S.: Causal confusion in imitation learning. arXiv preprint arXiv:1905.11979 (2019)
14. Khosla, A., Zhou, T., Malisiewicz, T., Efros, A.A., Torralba, A.: Undoing the damage of dataset bias. In: European Conference on Computer Vision, pp. 158–171. Springer (2012)
15. Krueger, D., Caballero, E., Jacobsen, J.H., et al.: Out-of-distribution generalization via risk extrapolation (rex). arXiv preprint arXiv:2003.00688 (2020)
16. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5542–5550 (2017)

17. Lin, L., Wen, L., Wang, J.: Mm-pred: A deep predictive model for multi-attribute event sequence. In: Proceedings of SDM, pp. 118–126. SIAM (2019)
18. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using convolutional neural networks for predictive process analytics. In: ICPM, pp. 129–136 (2019)
19. Piratla, V., Netrapalli, P., Sarawagi, S.: In: ICML (ed.) Efficient Domain Generalization via Common-Specific Low-Rank Decomposition, pp. 7728–7738. PMLR (2020)
20. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do imagenet classifiers generalize to imagenet? In: ICML, pp. 5389–5400. PMLR (2019)
21. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: International Conference on Machine Learning, pp. 1842–1850 (2016)
22. Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., Sarawagi, S.: Generalizing across domains via cross-gradient training. arXiv preprint arXiv:1804.10745 (2018)
23. Srivastava, M., Hashimoto, T., Liang, P.: In: In: ICML. (ed.) Robustness to Spurious Correlations via Human Annotations, pp. 9109–9119. PMLR (2020)
24. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30
25. Taymouri, F., Rosa, M.L., Erfani, S., Bozorgi, Z.D., Verenich, I.: Predictive business process monitoring via generative adversarial nets: the case of next event prediction. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 237–256. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_14
26. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. IEEE CVPR (2017)
27. Venkateswaran, P., Muthusamy, V., Isahagian, V., Venkatasubramanian, N.: Environment agnostic invariant risk minimization for classification of sequential datasets. In: Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, p. To appear (2021)