Explainability for Sequential Decision-Making

[Extended Abstract]

João Pedro Bento Sousa
joao.pedro.bento.sousa@tecnico.ulisboa.pt

Advisors: Prof. Dr. Mário Figueiredo and Dr. Pedro Saleiro

***Abstract*—Machine learning has been used to aid decision-making in several domains, from healthcare to finance. Understanding the decision process of ML models is paramount in high-stakes decisions that impact people's lives, otherwise, loss of control and lack of trust may arise. Often, these decisions have a sequential nature. For instance, the transaction history of a credit card must be considered when predicting the risk of fraud of the most recent transaction. Although RNNs are state-of-the-art models for many sequential decision-making tasks, they are perceived as black-boxes, creating a tension between accuracy and interpretability. While there has been considerable research effort towards developing explanation methods for ML, recurrent models have received relatively much less attention. Recently, Lundberg and Lee unified several methods under a single family of additive feature attribution explainers. From this family, KernelSHAP has seen a wide adoption throughout the literature; however, this explainer is unfit to explain models in a sequential setting, as it only accounts for the current input not the whole sequence. In this work, we present TimeSHAP, a model-agnostic recurrent explainer that builds upon KernelSHAP and extends it to sequences. TimeSHAP explains recurrent models by computing feature-, timestep-, and cell-level attributions, producing explanations at both the feature and time axes. As sequences may be arbitrarily long, we further propose two pruning methods that are shown to dramatically decrease TimeSHAP's computational cost and increase its reliability. We validate TimeSHAP by using it to explain predictions of two RNN models in two real-world fraud detection tasks, obtaining relevant insights into these models and their predictions.**

I. Introduction

With developments in *deep learning* (DL), the complexity of *machine learning* (ML) models has dramatically increased, resulting in a black-box paradigm in which data scientists and decision-makers (the end-users) strive to understand the decision processes of these models, hindering trust. Understanding the decision-making processes of complex models may also be crucial to detect and correct flawed reasoning [1] or possible discriminatory reasoning against specific sub-groups of the population, based on race, gender, or any other sensitive attribute [2], [3].

By explaining the reasoning in a given model, we simultaneously gather insight into how it may be improved and may advance human understanding of the underlying task. Moreover, regulators may want to be able to audit automated decision-making systems to make sure they are compliant with existing domain-specific regulations (e.g. finance). Understanding the model's reasoning may be a requirement in certain real-world applications, as exemplified in GDPR's "right to explanation" [4] (although its reach is contested [5], [6]).

This work focuses on sequential decision-making tasks, very common in several domains, such as healthcare and finance. In this kind of task, each decision takes into account the full history of events of the entity of interest (e.g., a patient or a bank account). *Recurrent neural network* (RNN) models, such as *long-short term memory* (LSTM) [7] and *gated recurrent unit* (GRU) [8], are state-of-the-art models in these domains. However, while their application has grown ubiquitous and its performance steadily increased, the complex decision-making processes in these models is seen as a black-box, creating a tension between accuracy and interpretability.

In recent years, numerous methods have been put forth to explain DL models [9]–[20]. However, RNNs pose a distinct challenge, as their predictions are not only a function of the immediate input instance, but also of the previous input instances in the sequence and the context (hidden state) drawn thereafter. Blindly applying state-of-the-art DL explainers to RNNs often disregards the importance of the hidden state, attributing all the feature importance solely to the features of the current input (as illustrated by Figure 1).

Recently, in landmark work, Lundberg and Lee [20] unified a large number of explanation methods into a single family of "additive feature attribution methods". The authors further proved that there is a unique solution to these attribution methods that fulfills three desirable properties of explanations (which will be described in detail below), namely *consistency*, *local accuracy*, and *missingness*, dubbing this implementation KernelSHAP. Although this method has become quite popular, it is not directly applicable to sequential decision-making tasks, as it only calculates attributions at the feature level and for a single input instance.

In this dissertation, we propose TimeSHAP, a post-hoc model-agnostic method to explain recurrent models. TimeSHAP leverages KernelSHAP's [20] strong theoretical foundations and empirical results, and adapts it to the recurrent setting. By doing so, we enable explaining, not only which features are most important to a recurrent model, but also which previous events had the largest impact on the current prediction.

As sequences are arbitrarily long, we further propose two pruning algorithms to dramatically reduce the computational cost of TimeSHAP by aggregating or removing older events, unimportant to the current prediction. By doing so, we improve

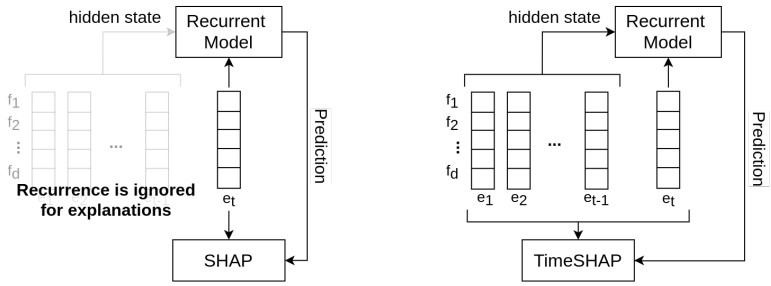

Fig. 1: Comparison between SHAP-based methods [20] from the literature (on the left) and TimeSHAP (on the right) when used to explain a recurrent model's predictions.

significantly the stability of TimeSHAP attributions and reduce the computational cost needed to obtain them.

In summary, the main contributions of this dissertation are the following:

- Adaptation of KernelSHAP [20] to the recurrent setting;
- Calculation of three types of explanations: Feature-, event- and cell-wise explanations;
- Two pruning methods that reduce the execution cost of TimeSHAP and increase explanation reliability;
- Validation of our method in two real-world fraud detection datasets.

Part of the work developed in this dissertation was accepted and presented at the **NeurIPS 2020 workshop HAMLETS (Human And Machine in-the-Loop Evaluation and Learning Strategies)**[1] in the paper *TimeSHAP: Explaining Recurrent Models through Sequence Perturbations*.

II. Background and Related Work

A. RNN Preliminaries

Although our method can be used to explain any sequence model, throughout this paper we will use the example of a vanilla RNN, as it is both simple and widely used. Other types of recurrent models that could be used include the LSTM [7], the GRU [8], or *conditional random field* (CRF) [21]. The predictions of a sequence model at a given time-step t are a function, not only of the current input $x^{(t)}$, but also of its previous inputs $x^{(t-1)}, x^{(t-2)}, \ldots, x^{(0)}$. For RNNs, this recurrence is achieved indirectly through a hidden state $\mathfrak{h}$ that aims to encode all relevant information from previous time-steps. The prediction of an RNN, $\hat{y}^{(t)} = f(x^{(t)}, \mathfrak{h}^{(t-1)})$, is given by [22]

$$a^{(t)} = b + W\mathfrak{h}^{(t-1)} + Ux^{(t)}, \tag{1}$$

$$\mathfrak{h}^{(t)} = \sigma(a^{(t)}), \tag{2}$$

$$o^{(t)} = c + V\mathfrak{h}^{(t)}, \tag{3}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}), \tag{4}$$

where b and c are learnable bias vectors, U, V, and W are learnable weight matrices, and σ is a nonlinearity (often chosen to be the hyperbolic tangent). Moreover, $a^{(t)}$ is known as the activation, $\mathfrak{h}^{(t)}$ as the hidden state, $o^{(t)}$ is the output, $\hat{y}^{(t)}$ is a vector of probabilities over the output, where $\hat{y}_i^{(t)} = \mathrm{e}^{o_i^{(t)}} / \sum_{j=1} \mathrm{e}^{o_j^{(t)}}$. If the RNN is the last layer of the model, then $\hat{y}^{(t)}$ is used directly as the prediction. Otherwise, $o^{(t)}$ is passed to the subsequent layers, and the prediction is given by the last layer in the forward-pass.

B. Related Work

Research on ML explainers can generally be subdivided into two categories: model-agnostic and model-specific.

Model specific explainers exploit characteristics of the model's inner workings or architecture to obtain more accurate explanations of its reasoning [23]. The task of explaining RNNs is often tackled by using attention mechanisms [24]–[27]. However, whether attention can in fact explain a model's behavior is debatable and a known source of controversy in the ML community [28]–[30].

DL models, in which RNNs are included, can also be explained using gradient-based methods. These explainers attribute a weight w_i to each feature, representing the importance, or saliency, of the i-th feature, based on the partial derivatives of the prediction function $f(x)$ with respect to the input x_i: $w_i = \left|\frac{\partial f(x)}{\partial x_i}\right|$ [10], [11], [31]. Another approach in the class of gradient-based methods redefines the backpropagation rules instead of utilizing the out-of-the-box gradient of the explained methods. These methods iteratively backpropagate a relevance value (which initially corresponds to the predicted score) over all the layers of the model until the input layer is reached and the relevance is distributed across all features. The contribution that reaches each input feature is its final feature attribution. These methods define the rules to distribute the relevance at each layer depending on the domain in which they are inserted [13]–[16]. However, when explaining sequential inputs, DL-specific methods focus on each input, leaving event relevance as a largely unexplored research direction. Finally, this approach is susceptible to noise and complexities that may affect the gradient, like the vanishing gradients problem [24], [32]. Regarding RNN-specific explainers [17]–[19] through neural inspection, these models do not create explanations as they only allow users to gain insights into the model behaviour.

[1]HAMLETS official webpage: https://hamlets-workshop.github.io/

At the same time, these explainers are inflexible with regards to the model's architecture. For instance, if the RNN is a building block of a larger DL model, preceded/succeeded by other types of layers, it is not the direct input to the RNN we want to explain but the input to the model.

Model-agnostic explainers are substantially more flexible as they can explain any architecture and can be applied to already trained architectures, possibly in production. These explainers generally rely on post-hoc access to a model's predictions under various settings, such as perturbations of its input [33]. A perturbation h_x of the input vector $x \in \mathbb{X}^m$ is the result of converting all values of a coalition of features $z \in \{0,1\}^m$ to the original input space $\mathbb{X}^m$, such that $z_i = 1$ means that a feature i takes its original value x_i, and $z_i = 0$ means that a feature i takes some uninformative background value b_i representing its removal. Hence, the input perturbation function h_x is given as follows:

$$h_x(z) = x \odot z + b \odot (\mathbf{1} - z), \tag{5}$$

where $\odot$ is the component-wise product. The vector $b \in \mathbb{X}^m$ represents an uninformative input sample, which is often taken to be the zero vector [34], $b = \mathbf{0}$, or to be composed of the average feature values in the input dataset [20], $b_i = \overline{x_i}$.

Lundberg and Lee [20] unified this and other explainers (both model-agnostic and model-specific) into a single family of "additive feature attribution methods". Moreover, the authors prove that there is a single solution to this family of methods that fulfills both local accuracy (the explanation model should match the complex model locally), missingness (features that are set to be missing should have no impact on the predictions), and consistency (if a feature's contribution increases, then its attributed importance should not decrease).

Those authors put forth KernelSHAP [20], a model-agnostic explainer that fulfills these three properties. KernelSHAP approximates the local behavior of a complex model f with a linear model of feature importance g, such that $g(z) \approx f(h_x(z))$. The task of learning the explanation model g is cast as a cooperative game where a reward ($f(x)$, the score of the original model) must be distributed fairly among the players ($i \in \{1, \ldots, m\}$, the features). The optimal reward distribution is given by the Shapley values formulation [35]. However, obtaining the exact Shapley values for all features would imply generating all possible coalitions of the input, $z \in \{0,1\}^m$, which scales exponentially with m, the number of features in the model. As this task is computationally intractable, KernelSHAP approximates the exact values by randomly sampling feature coalitions [36]. The authors further show that a single coalition weighing kernel, $\pi_x(z)$, and a single loss metric, $L(f, g, \pi_x)$, lead to optimal approximations of the Shapley values:

$$\pi_x(z) = \frac{(m-1)}{\binom{m}{|z|}\,|z|\,(m-|z|)}, \tag{6}$$

$$L(f, g, \pi_x) = \sum_{z \in \{0,1\}^m} \left[f(h_x(z)) - g(z)\right]^2 \cdot \pi_x(z), \tag{7}$$

where $|z|$ is the number of non-zero elements of z, and L is the squared error loss used for learning g.

Although adopted by the ML community, KernelSHAP is not applicable to sequential decision-making tasks, as it only calculates attributions at the feature level and for a single input instance, disregarding all previous sequence inputs. There have been a few approaches to extend this method to recurrent settings, but with debatable implementations. Ho et al. [37] use KernelSHAP to explain RNN predictions on an ICU mortality dataset [38]. However, their implementation perturbs only the time-step t being explained, ignoring all the previous elements of the sequence, and therefore only calculates attributions at the feature level for the one most recent feature vector, ignoring the sequential nature of the input.

III. TIMESHAP

The main goal of our work is to produce a recurrent local model explainer that is model-agnostic, post-hoc, and able to be applied in the recurrent setting. In order to explain the decision about one instance, our method should provide both event and feature attribution values. At the same time, this method should be resource-efficient to be applicable in real-world scenarios. Additionally, we aim to explain sequential models while preserving the three desirable properties of importance attribution stemming from the Shapley values: local accuracy, missingness, and consistency [39]. Hence, we put forth TimeSHAP, a model-agnostic recurrent explainer with sound theoretical footing and strong empirical results.

TimeSHAP builds upon KernelSHAP [20], a state-of-the-art model-agnostic explainer, and extends it to work on sequential data. Our method produces feature-wise and event-wise explanations. Hence, TimeSHAP attributes an importance value to each feature/event in the input that reflects the degree to which that feature/event affected the final prediction. In order to explain a sequential input, $X \in \mathbb{R}^{d \times l}$, with l events and d features per event, our method fits a linear explainer g that approximates the local behavior of a complex explainer f by minimizing the loss given by Equation 7. As events are simply features in a temporal dimension, and the algorithm for explaining features $x \in \mathbb{R}^{1 \times l}$ and events $x \in \mathbb{R}^{d \times 1}$ is conceptually equal, we will henceforth use the word *feature* to mean both rows and columns of $X \in \mathbb{R}^{d \times l}$. Thus, the formula for g is:

$$f(h_X(z)) \approx g(z) = w_0 + \sum_{i=1}^{m} w_i z_i, \tag{8}$$

where the bias term $w_0 = f(h_X(\mathbf{0}))$ corresponds to the model's output with all features toggled off (dubbed *base score*), the weights $w_i, i \in \{1, \ldots, m\}$, correspond to the importance of each feature, and either $m = d$ or $m = l$, depending on which dimension is being explained. The perturbation function $h_X : \{0,1\}^m \mapsto \mathbb{R}^{d \times l}$ maps a coalition $z \in \{0,1\}^m$ to the original input space $\mathbb{R}^{d \times l}$. Note that the sum of all feature importances corresponds to the difference between the model's score $f(X) = f(h_X(\mathbf{1}))$ and the base score $f(h_X(\mathbf{0}))$.

Input perturbations are generated differently, depending on which dimension is being explained. The perturbation function described in Equation 5 is suited to explain a single dimension of features. We extend this function to the recurrent (and bi-dimensional) setting as follows. Given a matrix $B \in \mathbb{R}^{d \times l}$ representing an uninformative input (the absence of discriminative features or events), a perturbation h^f_X along the features axis (the rows) of the input matrix $X \in \mathbb{R}^{d \times l}$ is the result of mapping a coalition vector $z \in \{0,1\}^d$ to the original input space $R^{d \times l}$, such that $z_i = 1$ means that row i takes its original value $X_{i,:}$, and $z_i = 0$ means that row i takes the background uninformative value $B_{i,:}$. Thus, when $z_i = 0$ the feature i is essentially toggled off for all events of the sequence. This is formalized by,

$$h^f_X(z) = D_z X + (I - D_z)B, \quad D_z = \text{diag}(z). \tag{9}$$

On the other hand, a perturbation h^e_X along the events axis (the columns) of the input matrix $X \in \mathbb{R}^{d \times l}$ is the result of mapping a coalition vector $z \in \{0,1\}^l$ to the original input space $R^{d \times l}$, such that $z_j = 1$ means that column j takes its original value $X_{:,j}$, and $z_j = 0$ means that column j takes the value $B_{:,j}$. Thus, when $z_j = 0$ all features of event j are toggled off. This is formalized by,

$$h^e_X(z) = XD_z + B(I - D_z), \quad D_z = \text{diag}(z). \tag{10}$$

Hence, when explaining features $h_X = h^f_X$, and when explaining events $h_X = h^e_X$. This change in the perturbation function is the sole implementation difference between explaining events and features. Moreover, the perturbation of X according to a null-vector coalition $z = \mathbf{0}$ is the same regardless of which dimension is being perturbed, $h^f_X(\mathbf{0}) = h^e_X(\mathbf{0})$, and equally for $z = 1$, $h^f_X(\mathbf{1}) = h^e_X(\mathbf{1})$.

In our setting, we define the background matrix $B \in \mathbb{R}^{l \times d}$ as containing the average feature values in the training dataset,

$$B = \begin{bmatrix} \overline{x_1} & \dots & \overline{x_1} \\ \overline{x_2} & \dots & \overline{x_2} \\ \vdots & \ddots & \vdots \\ \overline{x_l} & \dots & \overline{x_l} \end{bmatrix}. \tag{11}$$

A. Pruning

One glaring issue with TimeSHAP is that the number of event (temporal) coalitions scales exponentially with the length of the observed sequence, just as in KernelSHAP the number of feature coalitions scales exponentially with the number of input features. Moreover, in a recurrent setting, the input sequence can be arbitrarily long, making this a serious issue that we address by proposing two pruning algorithm.

In a real-world scenario, it is common for events to be preceded by a long history of past events, with only a few of them being relevant to the current prediction (e.g, the whole transaction history of a client to detect fraud on the most recent one). Additionally, recurrent models are known to seldom encode information from events in the distant past [40].

With the previously stated insight, we group together older (unimportant) events as a single coalition of events, thereby reducing the number of coalitions by a factor of 2^{l-i+1}, where i is the number of grouped events of a sequence with l elements. Essentially, we are sacrificing explanation granularity on older, probably unimportant, events, in favor of a runtime improvement and increased precision of explanations on important events. By grouping these events as one, we are allowing this group of events to still be considered in the explanations, receiving their, albeit smaller, importance. The pruning method, defined in Algorithm 1, consists in splitting the input sequence $X \in \mathbb{R}^{d \times l}$ into two sub-sequences $X_{:,1:i}$, $X_{:,i+1:l}$, $i \in \{1, \dots, l-1\}$ ($X_{:,l}$ being the most recent event), and computing the true Shapley values for each. Computing these Shapley values amounts to considering $2^2 = 4$ coalitions. Our objective is to find the largest i such that the importance value for $X_{:,1:i}$ falls below a given importance threshold η.

Alternatively, following the same rationale that older events are seldom important to the most recent prediction, we propose a sequence pruning algorithm. This method, described in Algorithm 2, is based on the intuition that if a sequence's score does not change significantly when adding events, then those events are likely unimportant to the decision-making process. As such, in order to prune an input sequence $X \in \mathbb{R}^{d \times l}$, we look for the smallest sub-sequence of recent events $X' = X_{:,i:l}, i \in \{1, \dots, l\}$, that matches the model's original score, within a given relative tolerance η, $|f(X) - f(X')| \leq f(X) \cdot \eta$. The number of coalitions is hence reduced by a factor of 2^{l-i+1}, where i is the number of grouped events of a sequence with l elements.

While Algorithm 1 prunes the temporal coalitions of a given input sequence maintaining the original sequence, Algorithm 2 prunes the sequence itself, changing the predicted score within a given error bound. The execution of both pruning algorithms scales only linearly with the number of events, $\mathcal{O}(l)$. Consequently, when employing pruning, the run-time of TimeSHAP is reduced from $\mathcal{O}(2^l)$ to $\mathcal{O}(2^{l-i})$ with i being the number of events lumped together. As older events are usually unimportant, a high number of events is pruned, thus $l - i \ll l$, rendering the performance increase highly significant. When considering the run-time of the underlying model, which for recurrent models scales at least linearly with the length of the input sequence, the two proposed pruning algorithms differ. Taking the best-case scenario of a model whose run-time scales linearly with l, such as a recurrent neural model (*e.g., RNN, LSTM, GRU*), TimeSHAP's run-time with temporal-coalition pruning (Algorithm 1) is $\mathcal{O}(l2^{l-i})$, while TimeSHAP's run-time with sequence pruning (Algorithm 2) is $\mathcal{O}((l-i)2^{l-i})$. Overall, Algorithm 2 leads to a faster execution, but with possibly lower fidelity explanations than Algorithm 1, as the explained sequence X' is a subsequence of the original input X.

B. Cell-level Explanations

Event-level and feature-level explanations provided by TimeSHAP indicate which columns (events) and features (rows) of

Algorithm 1 Temporal Coalition Pruning

Input: input sequence X,
model to explain f,
tolerance η,

for $i \in \{l-1, l-2, \ldots, 1\}$ **do** ▷ Starting from the end of the sequence
  $Z \leftarrow \{[0,0],[0,1],[1,0],[1,1]\}$ ▷ Full set of coalitions to use for each i
  $w_1, w_2 \leftarrow$ *KernelSHAP*(▷ Call adapted KernelSHAP
    model=f,
    input=$[X_{:,1:i}, X_{:,i+1:l}]$, ▷ X given as composed of only two features
    perturbation=h^e_X, ▷ Parameterized by our temporal perturbation function
    coalitions=Z) ▷ Employing only 2^2 coalitions (SHAP sees only 2 features)
  if $|w_1| < \eta$ **then** ▷ w_1 is the aggregate importance of all events up to i
    return i ▷ Index from which it is safe to lump event importances
return 0 ▷ No sequential group of events fits the pruning criteria

Algorithm 2 Score-based Sequence Pruning

Input: input sequence X,
model to explain f,
tolerance η,

$\hat{y} \leftarrow f(X)$
for $i \in \{l, l-1, \ldots, 1\}$ **do** ▷ Starting from the end of the sequence
  $X' \leftarrow X_{:,i:l}$ ▷ Truncate to most recent events
  if $|\hat{y} - f(X')| \leq \hat{y} \cdot \eta$ **then** ▷ If heuristic is fulfilled
    return X' ▷ Sequence pruned to $l - i + 1$ recent events
return X ▷ Sequence cannot be pruned

the input matrix are most relevant for the model prediction. We propose also a cell-level explanation method that indicates the most relevant features at particular events.

Cell-level explanations can be obtained by turning individual cells of the input matrix on and off. Although this approach calculates the individual contribution of each cell, it is intractable to compute, as the number number of coalitions is the product of the number of events with the number of features. Considering a small input example $X \in \mathbb{R}^{40\times 20}$, there exists 800 cells, thus $\mathcal{O}(2^{800})$ coalitions. In order to obtain reliable cell-level explanations, the number of cells to consider and perturb needs to be drastically reduced.

To obtain reliable cell-level explanations, TimeSHAP forms groups of cells calculating the attribution of these groups. Doing so reduces the granularity of the explanations, but makes its computation possible. In order to find the most relevant cells, TimeSHAP calculates both event- and feature-level explanations, indicating the most relevant features (rows) and events (columns) of the input matrix. Feature or events are considered relevant if their attribution value is higher than a user-defined threshold θ. Given that these rows and columns were signaled as relevant by TimeSHAP, it is assumed that the most, if not all, relevant cells of the input matrix are contained in the union of these rows and columns with special relevance to the cells on their intersection. As such, TimeSHAP isolates each cell of the intersection and calculates its individual attribution. After isolating the intersection cells, TimeSHAP then forms groups of cells that belong to the same relevant event/feature but are not on the intersection. Finally, TimeSHAP forms two additional cell groups: the cells belonging to the events lumped together by the temporal-coalition pruning, Algorithm 1, and all the remaining cells that do not belong to the union of the most relevant event/features and are not on the pruned events.

The aforementioned cell grouping strategy yields a good trade-off between cell granularity and computational cost. The number of considered cells with this strategy is $(|e||f|)+|e|+|f|+2$, where f and e represent the considered relevant features and events (contribution $> \theta$), while $|f|$ and $|e|$ denote their respective number.

Considering cell-level explanations with the grouping strategy defined before, coalitions of individual cells and cell groups are formed, represented by $z \in \{0,1\}^{(|e||f|)+|e|+|f|+2}$, where f and e represent the indices of the features and events considered relevant, and $|f|$ and $|e|$ are the corresponding lengths. As opposed to event-wise and feature-wise explanations, each element of a coalition vector z does not map directly into a column or a row. Instead, each $z_i \in z$ maps to a different cell group depending on its index i.

Calculating the cell-wise perturbation h^c_X of the input matrix $X \in \mathbb{R}^{d\times l}$ is the result of mapping a coalition vector z to the original input space $R^{d\times l}$, such that $z_i = 1$ means that cell group i takes its original value, and $z_i = 0$ means that cell group i takes the correspondent background uninformative value. Given a matrix $B \in \mathbb{R}^{l\times d}$, defined in Equation 11, the perturbation function h^c_X is then given by Algorithm 3.

Algorithm 3 Cell perturbation function

Input: coalition vector z, input sequence X, background matrix B, relevant events indices e, relevant features indices f, number of events d, number of features l, pruning index p

$cells \leftarrow permutations\_size\_2(e, f)$ ▷ Calculate intersection cells (event, feature)
$H \leftarrow B$ ▷ Start perturbation with all groups absent
for $i \leftarrow \{0, 1, \ldots, |z|\}$ **do** ▷ Iterate over coalition vector
  if $z[i] = 0$ **then** ▷ Cell group is turned off
    $continue$
  else if $i < |e| * |f|$ **then** ▷ Group is a cell
    $event, feature = cells[i]$ ▷ Obtain cell indices
    $H_{event,feature} = X_{event,feature}$ ▷ Turn on cell
  else if $i < (|e||f|) + |e|$ **then** ▷ Relevant event group
    $event = e[i - (|e||f|)]$ ▷ Obtain relevant event index
    $features = \{0, 1, \ldots, d\} \setminus f$ ▷ Obtain non-relevant features indices
    $H_{event,features} = X_{event,features}$
  else if $i < (|e||f|) + |e| + |f|$ **then** ▷ Relevant feature group
    $events = \{0, 1, \ldots, p\} \setminus e$ ▷ Obtain non-relevant event indices
    $feature = f[i - (|e||f|)]$ ▷ Obtain relevant feature index
    $H_{events,feature} = X_{events,feature}$
  else if $i < (|e||f|) + |e| + |f| + 1$ **then** ▷ Other non-pruned group
    $features = \{0, 1, \ldots, d\} \setminus f$ ▷ Obtain non-relevant features indices
    $events = \{0, 1, \ldots, p\} \setminus e$ ▷ Obtain non-relevant event indices
    $H_{events,features} = X_{events,features}$ ▷ Turn on other cells group
  else
    $features = \{0, 1, \ldots, d\}$ ▷ All feature indices
    $events = \{p + 1, \ldots, l\} \setminus e$ ▷ Non pruned events
    $H_{events,features} = X_{events,features}$ ▷ Turn on cell
return H ▷ No sequential group of events fits the pruning criteria

C. TimeSHAP Remarks

TimeSHAP computes explanations by fitting a linear explainer g that approximates the local behavior of a complex explainer f by minimizing the loss given in Equation 7. This linear explainer g is parameterized by our proposed perturbation functions, h^f_X, h^e_X, and h^c_X, depending on the desired explanations. To mitigate the exponential growth of the number of coalitions when explaining events, due to sequences being arbitrarily long, we further proposed two pruning algorithms (Algorithm 1 and Algorithm 2). These algorithms find the shortest relevant sequence input, as defined by the user, drastically reducing the number of considered coalitions or events, which in turn makes the produced explanations more reliable and reduces their computational cost. To further increase the reliability of cell-level explanations, we incorporate a cell grouping strategy into the cell perturbation function (defined in Algorithm 3).

IV. Experiments

To validate our method, we use it to explain predictions of two RNN models in two real-world fraud detection tasks. In this extended abstract, we will only present the results for one of them, due to space constraints. The results for the second dataset are present on the dissertation document.

The explained DL model is composed of an embedding layer for categorical variables, followed by a GRU layer [8], followed by a feed-forward classifier with four layers. The task for this dataset consists in predicting account takeover fraud, a form of identity theft where a fraudster gains access to a victim's bank account, enabling them to place unauthorized transactions. The data is tabular, consisting of approximately 20M events, including clients' transactions, logins, or enrollments[2], as well as corresponding geo-location and demographics data.

We run TimeSHAP on all sequences of the dataset that contain a positive prediction by the model. We set the maximum number of coalition samples to *n_samples* $= 32K$. Regarding the pruning algorithm, we employ our proposed temporal-coalition pruning algorithm and set its tolerance $\eta = 0.025$. On the full dissertation, we discuss in detail the rationale behind the selection of these parameters.

A. Pruning Results

In this Section we present the results from using the temporal-coalition pruning algorithm. Results for the sequence pruning algorithm are present on the full dissertation. For this experiment, we randomly select 1000 sequences and apply

[2]Examples of an *enrollment* event include changing the password or logging in from a new device.

our method, TimeSHAP to them, considering only event-level explanations, as feature-level explanations are not affected by the pruning.

Table I shows average, median, and maximum numbers of events for unpruned sequences, and for sequences pruned using Algorithm 1, with tolerance $\eta \in \{0.005, 0.0075, 0.01, 0.025, 0.05\}$. The percentage of sequences whose length, $|X|$, is smaller than $log_2(n\_samples) \approx 15$ is shown in the fourth row. This represents the percentage of sequences whose Shapley values can be exactly computed by exhaustively evaluating all $2^{|X|}$ coalitions. The Shapley values for all sequences longer than $log_2(n\_samples)$ are estimated by randomly sampling coalitions. We note that, for the original sequences, we can only compute exact Shapley values for 10% of the samples. For the median original sequence, with a total number of coalitions on the order of $2^{|X|} = 2^{139}$, 32000 sampled coalitions represents $10^{-36}\%$ of the total universe of coalitions. Hence, pruning is absolutely necessary in order to achieve accurate results. Using $\eta = 0.01$, we can compute exact Shapley values for 36.5% of sequences. On the other hand, using $\eta = 0.025$, we can compute exact values for 58.3% of the sequences. Finally, the last row of Table I shows the *relative standard deviation*[3] (RSD) [41] of the Shapley values over 10 random seeds; as expected, lower pruning tolerances lead to finer-grained event-level explanations (higher number of explained events) but with lower stability (higher RSD values). In fact, there is a strict negative relation between pruning tolerance and RSD values. Running TimeSHAP on the original sequences leads to very high variance (RSD 1.71), while the highest pruning tolerance $\eta = 0.05$ leads to relatively low variance (RSD 0.68).

B. Global Explanations

Supplying a data scientist with global explanations provides an overview of the model's decision process, revealing which features, or events in the case of TimeSHAP, are relevant to the model and which ones are not. This provides an insight into the decision process, guiding the data scientist's analysis of the model. TimeSHAP provides this type of explanations by explaining N sequences and drawing conclusions from aggregations and visualizations of the N local explanations.

Figure A.1 and A.2 present the event and feature global explanations for the *Account Takeover* (ATO) dataset produced by TimeSHAP applied to all sequences that contained a positive prediction, explaining the first positive prediction of each sequence. For hyperparameters, $\eta = 0.025$ and $n\_samples = 32000$ were used in the temporal coalition pruning algorithm.

Figure A.1 presents event-wise Shapley values (y-axis), for each event index (x-axis, with index 0 corresponding to the event being explained and decreasing indexes representing the distance to event 0). When evaluating event-wise explanations globally, it is possible to conclude that, on average, the transaction being explained, $t = 0$, is the event that most contributes to the score with an average contribution of 0.28. The next most relevant events, are those between indexes -1 and -4, with average contributions ranging from 0.05 to 0.13, indicating that the model is, on average, valuing these previous events to perform a decision at $t = 0$. For events with an index lower than -5, the average contribution is around 0. Nonetheless, it is possible to observe that there is a significant number of events between indexes -5 and -30 with high contributions. These significant contributions at distant events display the models' capability of keeping crucial information in the hidden state in order to compute the score of the event at $t = 0$, as well as the ability of TimeSHAP to capture it. We found that events at index $t = -1$ have a lower importance (0.05) than adjacent events, due to the fact that these events are, 80% of the time, logins that precede the explained transaction and, most of the times, these have a low attribution value given by TimeSHAP.

One crucial insight provided by these explanations is that the most recent event is responsible for, on average, 41% of the sequence's score, while the preceding events represent 59% of the sequence's score. This shows TimeSHAP's ability to understand the sequential domain, as other post-hoc methods would attribute 100% to the most recent event.

Figure A.2 in Appendix presents feature-wise Shapley values (x-axis) for each feature (indexed in the y-axis, ordered by mean attribution value). In this figure, feature names have been redacted due to privacy constraints. From Figure A.2, we conclude that some features have predominately positive contributions, having a higher average contribution than others. These features are, on average, valued by the model over other features in order to predict account takeover fraud and they consist of the Transaction (0.29) and Event (0.092) types, the clients' age (0.090), and features related to the IP and location of the events (0.08 to 0.03). These features, indicated as the most relevant ones by TimeSHAP, agree with domain knowledge where the event and transaction types together with IP and location features encode account behavior, while the age of the client might indicate, on average, the susceptibility of a person to fraud.

From Figure A.2, we also conclude that there are features that contribute, on average, negatively to the score. These features are related to the account (physical branch A and account feature D) and the type of token of the event in question. This features are related to the authentication and security of the account and the event itself. This negative contribution is in accordance with our domain knowledge, as these feature represent the authentication and security of the client and, therefore, provide confidence in the legitimacy of the event represented in the negative contribution to the score.

TimeSHAP's global analysis of feature explanation also allows data scientists to understand if there are features that are being ignored by the model. When analysing Figure A.2, two features, *IP Feature A* and *IP Feature B*, have null contribution for all explained sequences. Upon inspection of the raw data we see that these features take a single constant value for all

[3]A standardized measure of dispersion, computed as the ratio of the standard deviation to the mean, σ/μ.

	Original	$\eta = .005$	$\eta = .0075$	$\eta = .01$	$\eta = .025$	$\eta = .05$
Average seq. length	182.1	69.0	58.4	50.0	32.9	19.7
Median seq. length	138.5	33.0	27.0	23.0	14.0	9.0
Max seq. length	2187	2171	1376	1132	1130	879
Percentile at $log_2(32000)$	10.0	27.3	32.7	36.5	58.3	78.8
TimeSHAP RSD, $\frac{\sigma}{\mu}$	1.71	1.19	1.17	1.09	0.98	0.68

TABLE I: Temporal-coalition pruning analysis (Algorithm 1) for the ATO dataset. Sequence length indicates the number of events to be explained (events that were aggregated after pruning count as 1).

sequences. This demonstrates the usefulness of TimeSHAP in aiding the detection of useless or redundant features.

C. Local explanations

Local explanations explain the model's rationale regarding one specific instance. These explanations can be utilized in several use-cases, for example, for bias auditing or model debugging. However, these explanations can mostly be used by fraud analysts in order to aid on their decision task.

In this Section we present two case-studies/sequences. For the two analyzed sequences, we selected two predicted-positive sequences, hence labeled A and B. Sequence A has a model score of $f(A) = 0.57$, and a total length of 47 events. Sequence B has a model score of $f(B) = 0.84$, and a total length of 286 events. As a convention, we dub the current event's index (the most recent) as $t = 0$, and use negative indices for previous events (the event at index $t = -1$ immediately precedes the event at index $t = 0$, and so on).

Figures 2a and 2d show the Shapley values, (importance,) respectively, for Sequences A and B, split into two disjoint sub-sequences at a given index t. This corresponds to the application of the "*for loop*" in Algorithm 1, continuing even after the pruning condition has been fulfilled. Figure 2d only displays the first 100 indexes as displaying all 286 events would clutter the Figure. As expected, the aggregate importance of older events (from the beginning of the sequence up to index t) suffers a steep decrease as its distance to the current event increases. This trend corroborates our hypothesis and supports our coalition pruning algorithm. Using coalition pruning tolerance $\eta = 0.025$, Sequence A is pruned to 11 events, grouping the 36 older events together. Similarly, sequence B is pruned to 9 events, grouping the last 277 events together.

Event-wise explanations for sequence A are shown in Figure 2b, and its feature-wise explanations in Figure 2c. We conclude that there are two events crucial for the model's prediction: the transaction being explained ($t = 0$), with a Shapley value of 0.36, and another transaction, 4 events before ($t = -4$), with a Shapley value of 0.17. Between the two relevant transactions (events $-3 \leq t \leq -1$), there are three logins with little to no importance. Prior to event $t = -4$, there are 5 logins and one transaction with reduced importance, which were nonetheless left unpruned by Algorithm 1. Regarding feature importances, we observe that the most relevant features are, in decreasing order of importance, the transaction's *amount*, *IP feature D*, and the clients' *age*. When inspecting the raw feature data, we observe that the amount transferred at transaction $t = 0$ is unusually high, a known account takeover indicator. This is in accordance with the simultaneous high event importance for $t = 0$, together with the high feature importance for the transaction amount. Moreover, we observe that the client's age is relatively high, another well-known fraud indicator, as elderly clients are often more susceptible to being victims of fraud [42]. When analysing *IP feature D*, although this feature does not show any strange behavior, it assumes a value that is frequent throughout the dataset. Upon further inspection, we conclude that the IP belongs to a cloud hosting provider, which domain experts confirm to be suspicious behavior. Analyzing the cell-level explanations for this sequence, with $\theta = 0.1$, we verify the previously stated insight, where the high transaction amount at $t = 0$ is abnormal. This cell is solely responsible for 0.21 of the score, revealing an extreme importance in a sequence with more than 2000 cells. These cell-level explanations also show that the relevant cells are present on the relevant events, with the cells in all non-relevant events receiving a total importance of 0.005.

Event-wise explanations for sequence B are shown in Figure 2e, and its feature-wise explanations in Figure 2f. Regarding event importance, we conclude that the most relevant events are at indices -4 and -1, with their corresponding Shapley values of 0.48 and 0.24, followed by events -2 (0.089) and -3 (0.049). Interestingly, for this sequence, the most relevant event is not the current one ($t = 0$), with near null contribution to the score (0.001). The event types for the sequence of events from $t = -4$ to $t = -2$ are *enrollment-login-transaction*, a well-known pattern that is repeated on numerous stolen accounts. This sequence of events encodes a change of account settings, e.g., a password change (enrollment), followed by a login into the captured account, subsequently followed by one or more fraudulent transactions. Interestingly, events $t = -1$ and $t = -2$ are transactions that succeed the fraudulent enrollment and login, but precede the current transaction ($t = 0$). The information up to $t = -1$ is already sufficient for the model to correctly identify the account as compromised, corroborated by the low contribution of the transaction at $t = 0$.

Regarding feature importances, the most relevant features are related to the *transaction type*, *event type*, the clients' *age*, and the *location*. The feature *transaction type* indicates a finer grained event taxonomy for when *event type* = "transaction". When inspecting the raw feature data, we observe that the client is in the elderly age range, which, as previously mentioned, may indicate a more susceptible demographic. When analysing

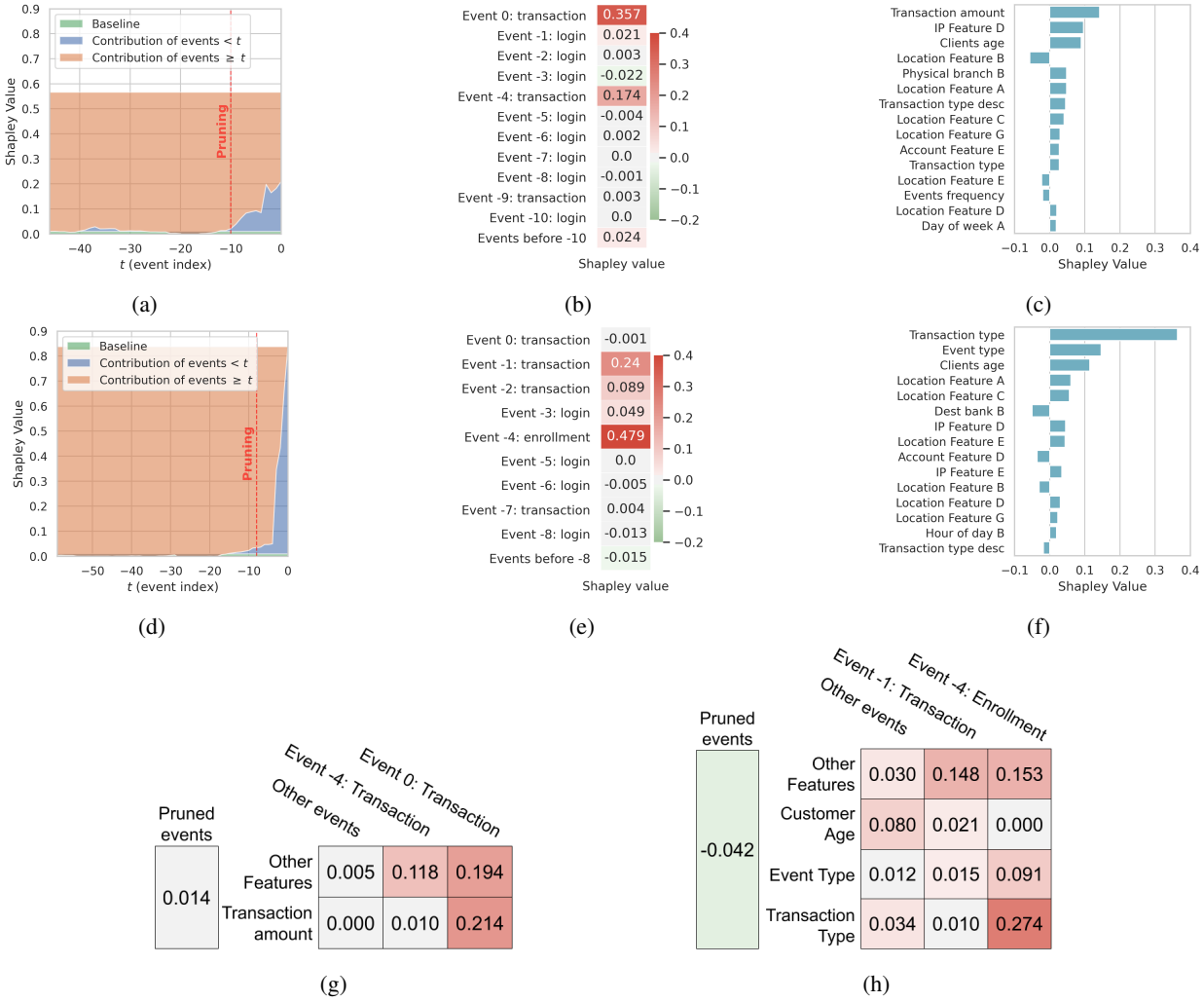

Fig. 2: Figures (a), (b), (c), and (g) show TimeSHAP results for Sequence A. Figures (d), (e), (f), and (h) show TimeSHAP results for Sequence B. Figures (a) and (d) show the importance of older ($X_{:,:t-1}$) vs current events ($X_{:,t:}$) calculated using Algorithm 1 also displaying also Shaps' local accuracy property. Event-level importance shown in Figures (b) and (e). Feature-level importance shown in Figures (c) and (f). Cell-level importance shown in Figures (g) and (h).

the location features *Location feature A* and *Location feature D*, we observe a discrepancy between the location of the enrollment, login and transactions from the account's history. This discrepancy in physical location is highly suspicious and indicates that there was an enrollment on the account from a previously unused location. Regarding the cell-level explanations, with $\theta = 0.1$, we obtain that the most relevant cell, with importance 0.274, is the intersection between the most relevant event and feature. This cell is followed by the groups of cells that aggregate the other features of the relevant events (separately) with the other features of event $t = -4$, having a relevance of 0.153, and of event $t = -1$, with an importance of 0.148. Considering the second most relevant feature, *Event type*, it appears to be relevant at event $t = -4$ with importance 0.091 and *Customer age* being important on mostly other non-relevant events. Another relevant insight provided by these explanations is that the cells of non-relevant features or events have an importance of 0.03.

V. Conclusion

Several sensitive domains have seen the application of complex ML models supporting decision-making tasks. From healthcare to finance, several of these domains present a sequential nature, where each decision is based, not on a single input, but on a sequence. RNNs have been applied to sequential domains, as they are state-of-the-art on these tasks. However, their complex decision-making process is still seen as a black-box, generating apprehension regarding their use in sensitive domains. While considerable effort has been guided towards explaining deep learning models, recurrent models have received relatively much less attention, with the majority of proposed methods being unfit for the sequential setting. Recently, the SHAP [20] framework unified several methods under a single family of additive feature attribution explainers.

From this family, the model-agnostic explainer, KernelSHAP, has seen a wide adoption throughout the literature, however, this explainer is unfit to explain models in a sequential setting, as it only accounts for the input at hand instead of the whole sequence.

In this dissertation, we proposed TimeSHAP, a post-hoc model-agnostic recurrent explainer. Our method leverages KernelSHAP's sound theoretical footing and strong empirical results and applies it to the sequential domain. Applying TimeSHAP to a recurrent model allows a user to obtain feature-level and event-level explanations. The explanations indicate the most relevant features throughout the sequence and the most relevant events/timesteps to the explained transaction, respectively. Given event and feature explanations, TimeSHAP also computes cell-level attribution indicating the most relevant features at particular events. In addition to the proposed explanation methods, we proposed two pruning algorithms that both decrease the execution time of TimeSHAP and increase the reliability of the explanations.

In order to validate TimeSHAP, we applied it to two real-world fraud decision tasks. For each task, we analyzed two individual case-studies displaying the TimeSHAP's explanation process. We also performed global analysis of our models, validating the obtained explanations with domain knowledge. Through this analysis, we were able to identify sequences where the most relevant event was several events previous to the event being explained and where the event being explain did not contribute to the score at all. We also showed that the most recent event of a sequence is, on average, responsible for 41% of the sequence's score, while the preceding events represent 59% of the sequence's score. TimeSHAP also provided the insight that, for the ATO dataset, one of the most relevant features was the age of the client, hinting at potentially discriminatory reasoning. This insight was confirmed after performing a bias-audit on our model, showcasing TimeSHAP's capabilities to find possible bias vectors.

Finally, the main contributions of this work can be summarized as follows:

- We adapted KernelSHAP to the recurrent setting, redefining it's perturbation functions, background instance, and other necessary adaptations;
- We proposed a method to calculate three types of attributions: feature-, event- and cell-wise explanations;
- We propose two difference pruning algorithms to reduce the computational cost of the method and increase explanation reliability;
- We performed an empirical analysis and validation of our method on two real-world fraud detection datasets;

VI. Future Work

For future work, we intend to further validate our method with users at Feedzai. We want to have data-scientists use TimeSHAP on their model development cycles in order to understand the impact of our method on their daily tasks. Additionally to data scientists, we intent to perform user testing with fraud analysts. This would quantify the usefulness of our method as a tool to help in the decision process. Moreover, regarding the evaluation of our method, we would like to test it against other methods in the literature that are being used to explain recurrent models.

Regarding TimeSHAP itself, we aim to develop the cell-level attributions even further in order to decouple them from both event and feature explanations.We also aim to develop a method equivalent to our sequence pruning but applied to the feature-axis. Although this has a fixed number of elements, there might exist models with large numbers of features, reducing the reliability of explanations. Finally, we want to explore different perturbation strategies based on domain knowledge, in order to achieve higher granularity on important attributes at the cost of human power.

References

[1] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, dec 2019.

[2] Megan Garcia. Racist in the machine: The disturbing implications of algorithmic bias. *World Policy Journal*, 33(4):111–117, 2016.

[3] James Zou and Londa Schiebinger. Ai can be sexist and racist-it's time to make it fair, 2018.

[4] General Data Protection Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88):294, 2016.

[5] Bryce Goodman and Seth Flaxman. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Magazine*, 38(3):50–57, oct 2017.

[6] Andrew Selbst and Julia Powles. "Meaningful information" and the right to explanation. In *Conference on Fairness, Accountability and Transparency*, pages 48–48. PMLR, 2018.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics.

[9] K. Simonyan, A. Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[10] Misha Denil, Alban Demiraj, and Nando De Freitas. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*, 2014.

[11] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, June 2016. Association for Computational Linguistics.

[12] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[13] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211 – 222, 2017.

[14] S. Lapuschkin, A. Binder, G. Montavon, K. Müller, and W. Samek. Analyzing classifiers: Fisher vectors and deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2912–2920, 2016.

[15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

[16] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153, 2017.

[17] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018.

[18] W. James Murdoch and Arthur Szlam. Automatic rule extraction from long short term memory networks, 2017.

[19] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[20] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[21] John D Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

[22] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-Agnostic Interpretability of Machine Learning. *ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, jun 2016.

[24] Aya Abdelsalam Ismail, Mohamed Gunady, Luiz Pessoa, Hector Corrada Bravo, and Soheil Feizi. Input-cell attention reduces vanishing saliency of recurrent neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10814–10824. Curran Associates, Inc., 2019.

[25] Jinghe Zhang, Kamran Kowsari, James H. Harrison, Jennifer M. Lobo, and Laura E. Barnes. Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record. *IEEE Access*, 6:65333–65346, 2018.

[26] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512. Curran Associates, Inc., 2016.

[27] Ying Sha and May D. Wang. Interpretable predictions of clinical outcomes with an attention-based recurrent neural network. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics*, ACM-BCB '17, page 233–240, New York, NY, USA, 2017. Association for Computing Machinery.

[28] Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics.

[29] Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[30] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, November 2019. Association for Computational Linguistics.

[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

[32] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[33] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018.

[34] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[35] H P Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72, jun 1985.

[36] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, dec 2014.

[37] Long V. Ho, Melissa D. Aczon, David Ledbetter, and Randall Wetzel. Interpreting a recurrent neural network's predictions of icu mortality risk, 2020.

[38] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[39] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[40] Y. Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3, 1993.

[41] Charles E. Brown. *Coefficient of Variation*, pages 155–157. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[42] Jinkook Lee and Horacio Soberon-Ferrer. Consumer vulnerability to fraud: Influencing factors. *The Journal of Consumer Affairs*, 31(1):70–89, 1997.

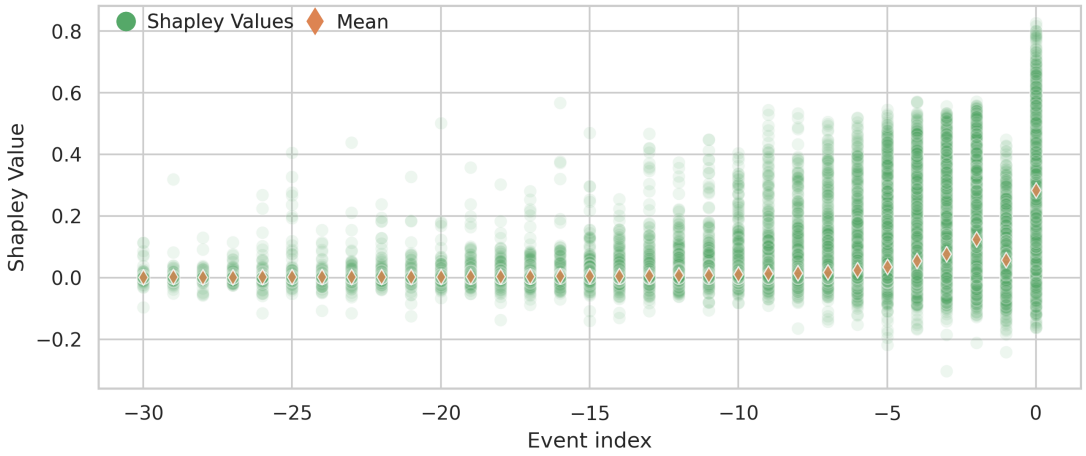

Fig. A.1: Event global explanations.

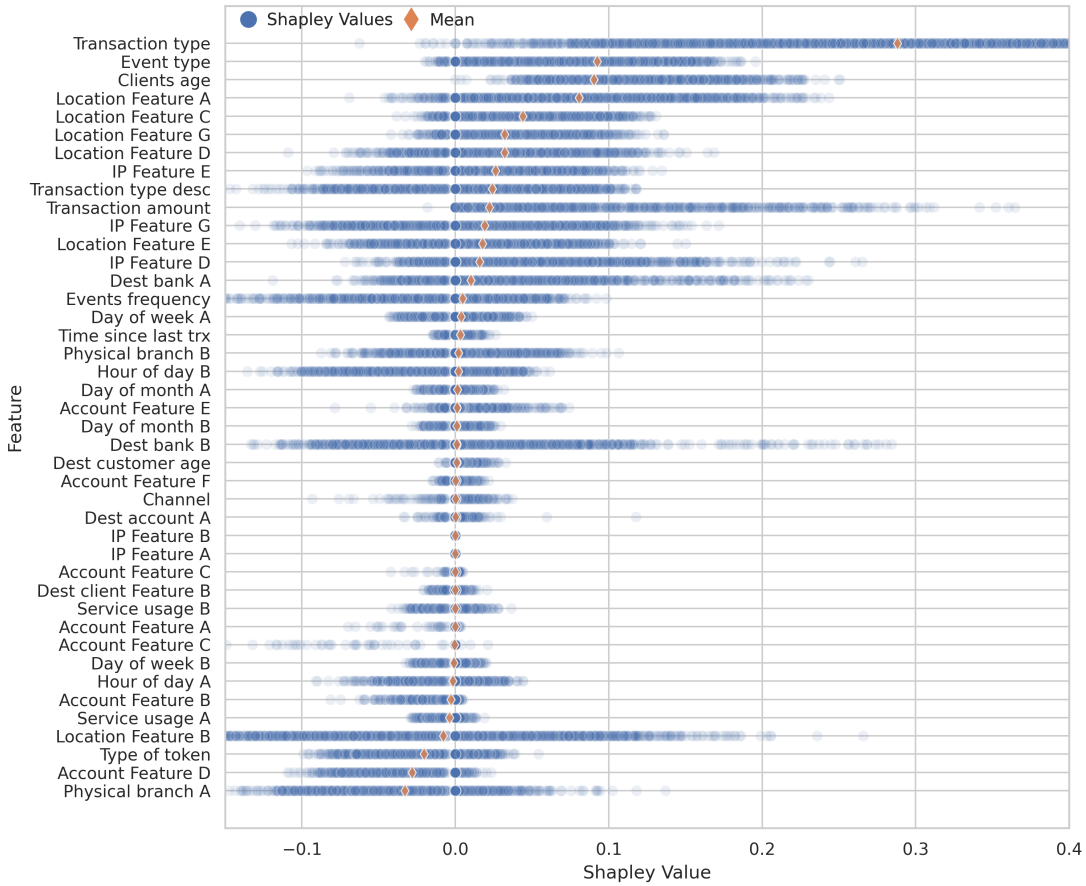

Fig. A.2: Feature global explanations.