TimeSHAP: Explaining Recurrent Models through Sequence Perturbations

João Bento[1,2] **Pedro Saleiro**[1] **André F. Cruz**[1]

Mário A.T. Figueiredo[2,3] **Pedro Bizarro**[1]

{joao.bento, pedro.saleiro, andre.cruz, pedro.bizarro}@feedzai.com
mario.figueiredo@tecnico.ulisboa.pt

[1] Feedzai
[2] LUMLIS (Lisbon ELLIS Unit), Instituto Superior Técnico, Lisbon, Portugal
[3] Instituto de Telecomunicações, Lisbon, Portugal

Abstract

Recurrent neural networks are a standard building block in numerous machine learning domains, from natural language processing to time-series classification. While their application has grown ubiquitous, understanding of their inner workings is still lacking. In practice, the complex decision-making in these models is seen as a black-box, creating a tension between accuracy and interpretability. Moreover, the ability to understand the reasoning process of a model is important in order to debug it and, even more so, to build trust in its decisions. Although considerable research effort has been guided towards explaining black-box models in recent years, recurrent models have received relatively little attention. Any method that aims to explain decisions from a sequence of instances should assess, not only feature importance, but also event importance, an ability that is missing from state-of-the-art explainers. In this work, we contribute to filling these gaps by presenting TimeSHAP, a model-agnostic recurrent explainer that leverages KernelSHAP's sound theoretical footing and strong empirical results. As the input sequence may be arbitrarily long, we further propose a pruning method that is shown to dramatically improve its efficiency in practice.

1 Introduction

Recurrent neural networks (RNN) models, such as LSTMs [1] and GRUs [2], are state-of-the-art for numerous sequential decision tasks, from speech recognition [3] to language modelling [4] and time-series classification [5]. However, while the application of RNNs has grown ubiquitous and its performance has steadily increased, understanding of its inner workings is still lacking. In practice, the complex decision-making processes in these models is seen as a black-box, creating a tension between accuracy and interpretability.

Understanding the decision-making processes of complex models may be crucial in order to detect and correct flawed reasonings, such as those stemming from spurious correlations in the training data. Models that rely on such spurious correlations, known as "Clever Hans" models[1] [7], may have

[1]Named after an early 20th century horse that was thought to be able to do arithmetic, but was later found to be picking up on behavioral cues from its owner when pointing to the correct answer [6].

Preprint. Under review.

strong test results but generalize poorly when deployed in the real-world. 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, as previously unknown patterns are uncovered by the explanation.

Additionally, understanding the model's reasoning may be a requirement in certain real-world applications, as exemplified in GDPR's "right to explanation" [8] (although its reach is contested [9, 10]). Just as humans are biased and sometimes discriminatory towards certain groups of the population, so too can *deep learning* (DL) models be [11, 12, 13]. Regulators want to be able to peek under the "model's hood" in order to audit for potential discriminatory reasoning. Although humans may be less accurate, and certainly less scalable than DL models, they can offer some form of after-the-fact reasoning supporting their decisions. For all the benefits and efficiencies DL has brought about, in order for the community to trust these models, it must be possible to explain their reasoning, at least to a level that humans can understand.

In recent years, numerous methods have been put forth for explaining DL models [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. However, RNNs pose a distinct challenge, as their predictions are not only a function of the immediate input sample, but also of the previous input samples 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, distributing all feature importance solely through the features of the current input (as illustrated by Figure 1).

Recently, in landmark work, Lundberg and Lee [25] 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 this task that fulfills three crucial properties of explanations, and dubbed this implementation KernelSHAP. However, no work yet extended this method to explaining timestep-wise RNN predictions.

With this in mind, we propose TimeSHAP, a model-agnostic recurrent explainer suited for tabular sequence learning that leverages KernelSHAP's strong theoretical foundations and empirical results.

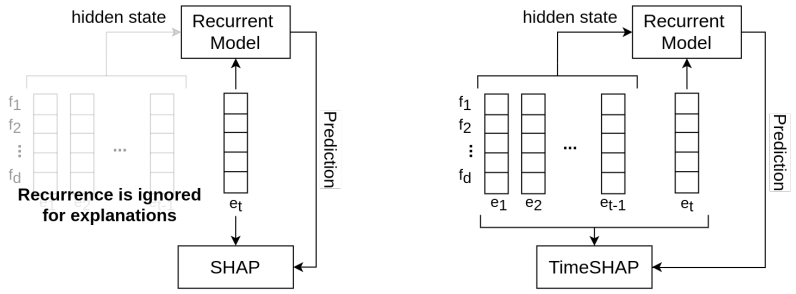

Figure 1: Comparison between SHAP-based methods from the literature (on the left) and TimeSHAP (on the right) when used to explain a recurrent model's predictions.

TimeSHAP extends the framework put forth by Lundberg and Lee [25] to the recurrent model setting. By doing so, we enable explanation, not only of which features are most important to a recurrent model, but also which previous events had the largest impact on a current prediction. As sequences are arbitrarily long, we further propose a pruning method that increases real-world accuracy and efficiency considerably. We analyze local and global explanations of an RNN using our method, and find multiple instances in which these are crucial for debugging the underlying predictor.

The contributions of TimeSHAP can be summarized as follows:

- explanations of both feature- and event-wise importance in sequence predictions;
- a new perturbation function suited for the recurrent setting;
- a coalition pruning algorithm that dramatically increases the method's efficiency in practice;
- an empirical analysis of our method on a real world banking dataset.

2 Related Work

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

Model specific explainers exploit characteristics of the model's inner workings or architecture to obtain more accurate explanations of its reasoning [26]. The task of explaining RNNs is often tackled by using attention mechanisms [27, 28, 29, 30]. However, whether attention can in fact explain a model's behavior is debatable and a known source of controversy in the ML community [31, 32, 33].

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: $w_i = \left|\frac{\partial f(x)}{\partial x_i}\right|$ [14, 15, 16]. Another family of DL explainers is that of layer-wise relevance propagation. In the network's backward pass, starting from the output neuron, the relevance (which initially corresponds to the predicted score) is propagated iteratively from higher layers to lower layers, according to some rules [18, 19, 20, 21]. However, when explaining sequential inputs, DL-specific methods focus on features instead of events, leaving event relevance as a largely unexplored research direction. Regarding RNN-specific explainers [22, 23, 24], these are often 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. Hence, a model-agnostic explainer may be better suited to explain these architectures, as real-world models are seldom built only with recurrent layers.

Model-agnostic explainers are substantially more flexible, thus often preferred in real-world applications. These explainers generally rely on post-hoc access to a model's predictions under various settings, such as perturbations of its input [34]. 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:

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

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 [35], $b = \mathbf{0}$, or to be composed of the average feature values in the input dataset [25], $b_i = \overline{x_i}$.

Lundberg and Lee [25] 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, 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 [36]. However, obtaining the true 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, $\mathcal{O}(2^m)$. As this task is computationally intractable, KernelSHAP approximates the true values by randomly sampling feature coalitions [37]. 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:

$$\pi_x(z) = \frac{(m-1)}{\binom{m}{|z|}\,|z|\,(m-|z|)} \tag{2}$$

$$L(f, g, \pi_x) = \sum_{z \in \{0,1\}^m} [f(h_x(z)) - g(z)]^2 \cdot \pi_x(z) \tag{3}$$

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

Despite being widely adopted by the ML community, KernelSHAP cannot be applied out-of-the-box to recurrent models. Doing so disregards an important part of the model: the hidden state, which carries information on the previous predictions. Hence, the current KernelSHAP framework cannot be applied to recurrent models due to the following shortcomings: (1) assuming independence between subsequent model predictions, and (2) disregarding the importance of the hidden state $\mathfrak{h}$ as a feature. There have been few approaches to extend this method to recurrent settings, but with debatable implementations. Ho et al. [38] use KernelSHAP to explain RNN predictions on an ICU mortality dataset [39]. However, their implementation perturbs only the time-step t being explained (shortcoming (1)), and distributes the model's score $f(x^{(t)}, \mathfrak{h}^{(t-1)})$ solely through the input features $x^{(t)}$ (shortcoming (2)).

3 Methodology

Firstly, our goal is to enable explaining sequence models while preserving three desirable properties of importance attribution stemming from the Shapley values: local accuracy, missingness, and consistency [40]. Secondly, as sequence models handle one extra dimension (spanning the sequence of input events), we aim to explain both feature importance and event importance. Finally, our method should be resource-efficient, as calculating the true Shapley values is trivial but computationally intractable. Hence, we put forth TimeSHAP, a model-agnostic recurrent explainer with sound theoretical footing and strong empirical results.

3.1 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 long short-term memory (LSTM) [1], the gated recurrent unit (GRU) [2], or conditional random field (CRF) [41]. 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 RNN, this recurrence is achieved indirectly through a hidden state $\mathfrak{h}$ that aims to encode all relevant information from previous time-steps. As such, an RNN's prediction, $\hat{y}^{(t)} = f(x^{(t)}, \mathfrak{h}^{(t-1)})$, is given as follows [42]:

$$a^{(t)} = b + W\mathfrak{h}^{(t-1)} + Ux^{(t)}, \tag{4}$$

$$\mathfrak{h}^{(t)} = \sigma(a^{(t)}), \tag{5}$$

$$o^{(t)} = c + V\mathfrak{h}^{(t)}, \tag{6}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}), \tag{7}$$

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, and $\hat{y}$ is a vector of probabilities over the output. If the RNN is the last layer of the model, then $\hat{y}^{(t)}$ may be used directly as the prediction. Otherwise, $o^{(t)}$ is passed to the following layers, and the prediction is given by the last layer in the forward-pass.

3.2 TimeSHAP

TimeSHAP builds upon KernelSHAP [25], a state-of-the-art model agnostic explainer, and extends it to work on sequential data. Our method produces both feature-wise and event-wise explanations. Hence, TimeSHAP attributes an importance value to each feature/event in the input, such that it 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 3. 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:

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

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 1 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'_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 as follows:

$$h'_X(z) = D_z X + (I - D_z)B, \quad D_z = \mathrm{diag}(z). \tag{9}$$

On the other hand, a perturbation h^*_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 as follows:

$$h^*_X(z) = X D_z + B(I - D_z), \quad D_z = \mathrm{diag}(z). \tag{10}$$

Hence, when explaining features $h_X = h'_X$, and when explaining events $h_X = h^*_X$. 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, $h'_X(\mathbf{0}) = h^*_X(\mathbf{0})$, and equally for $z = 1$, $h'_X(\mathbf{1}) = h^*_X(\mathbf{1})$.

In our setting, we define the background matrix $B \in \mathbb{R}^{l \times d}$ to be composed of the average feature values in the training dataset:

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

3.2.1 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 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 a temporal-coalition pruning algorithm.

It is common for real-world events to be preceded by a long history of past events (e.g., the whole transaction history of a client), only a few of which are relevant to the current prediction. Additionally, recurrent models are known to seldom encode information from events in the distant past [43]. Based on this insight, we group together older unimportant events as a single feature, thereby reducing the number of coalitions by a factor of 2^{i-1}, where i is the number of grouped events. Essentially, we lose the granularity between the importance of these grouped events, but their importance (albeit small) is not disregarded.

The pruning method, defined in Algorithm 1, consists in splitting the input sequence $X \in \mathbb{R}^{d \times l}$ into two sub-sequences $X_{:,1:i}$, $X_{:,i+1:l}$, $i \in \{1, \ldots, l-1\}$, ($X_{:,l}$ being the most recent event) and computing the true Shapley values for each. Computing these Shapley values amounts to $2^2 = 4$ total coalitions (for each i). Our objective is to find the largest i such that the importance value for $X_{:,1:i}$ falls below a given importance threshold η.

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^*_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

The computational cost of this pruning algorithm scales only linearly, $\mathcal{O}(l)$, with the number of events. Consequently, when employing pruning, the run-time of TimeSHAP when explaining the events' axis is reduced from $\mathcal{O}(2^l)$ to $\mathcal{O}(2^{l-i})$. We will empirically show in Section 4.1 that events in the distant past seldom affect the model's score, leading to $l - i \ll l$. Taking the best-case scenario of a recurrent 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 totals $\mathcal{O}(l \cdot 2^{l-i})$. On the other hand, when explaining the features' axis, TimeSHAP's computational complexity is unaffected by the pruning algorithm, totalling $\mathcal{O}(l \cdot 2^d)$.

3.2.2 Algorithm

TimeSHAP's objective is to answer the following questions: "Which events and features contributed the most to the current prediction?" and "What was their influence on the model's score?" Hence, when explaining the features axis, TimeSHAP fits a linear explainer g that approximates the local behavior of a complex explainer f by minimizing the loss given in Equation 3, parameterized by our perturbation function h'_X. On the other hand, when explaining the events' axis, TimeSHAP first (optionally) prunes the input sequence's coalitions (as per Algorithm 1), and then fits the linear explainer, parameterized by our perturbation function h^*_X. Note that, as the number of coalitions scales exponentially with the sequence's length on the axis that is being explained (reduced by the pruning factor), it may not be tractable to exhaustively evaluate all coalitions. In this case, similarly to KernelSHAP, we randomly sample coalitions from the pool of coalitions up to a predetermined number of draws.

4 Experiments

To validate our method, we trained a recurrent deep learning model on a large-scale real-world banking dataset. The model is composed of an embedding layer for categorical variables, followed by a GRU layer [2], and subsequently followed by a feed forward layer The task consists in predicting account takeover fraud, a form of identity theft where a fraudster gains access to a victim's bank account,

Table 1: Temporal-coalition pruning analysis (Algorithm 1). Sequence length indicates the number of events to be explained (events that were aggregated after pruning count as 1). Relative standard deviation (RSD) of Shapley values computed over 10 runs of TimeSHAP.

	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 $log_2(32K)$	10.0	27.3	32.7	36.5	58.3	78.8
TimeSHAP RSD, $\frac{\sigma}{\mu}$	1.23	0.70	0.63	0.54	0.23	0.15

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 $1K$ randomly chosen sequences that were predicted positive by the model. We set the maximum number of coalition samples to *n_samples* $= 32K$. Regarding pruning, we employ our proposed temporal-coalition pruning algorithm, as it promotes exponentially faster execution at no cost to the explanation's reliability (only with decreased granularity on unimportant older events). For sequences with a number of events/features higher than log_2(*n_samples*), pruning does not impact performance directly, improving instead the accuracy of results, as longer sequences would need a higher number of coalition samples to accurately compute Shapley values. We can choose a pruning tolerance value that enables exhaustively computing the Shapley values for most input sequences within the allocated *n_samples* budget.

4.1 Pruning method results

Table 1 details average, median, and maximum number of events for unpruned sequences, and for sequences pruned with varying tolerance levels for Algorithm 1. The percentage of sequences whose length, $|X|$, is under log_2(*n_samples*) ≈ 15 is shown in the fourth row. This represents the percentage of input 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 $2^{|X|} = 2^{139}$, 32K sampled coalitions represents $10^{-36}\%$ of the total universe of coalitions. Hence, pruning is not only resource-efficient but also a necessary step in order to achieve accurate results.

When using $\eta = 0.01$, we can compute exact Shapley values for 36.5% of the input samples. On the other hand, when using $\eta = 0.025$, we can compute exact values for 58.3% of the input samples. Hence, we choose $\eta = 0.025$ as our pruning tolerance, providing a balance between explanations' consistency, run-time, and granularity.

The last row of Table 1 shows the relative standard deviation[3] [44] (RSD) of the Shapley values obtained over 10 runs of TimeSHAP for different pruning levels. As expected, lower pruning tolerances (lower η) lead to finer-grained event-level explanations (higher number of explained events) but with lower reliability (higher RSD values). In fact, there is a strict negative relation between pruning tolerance and RSD values. Running TimeSHAP on the original sequences (equivalent to $\eta = 0$) leads to very high variance (RSD 1.23), while the highest pruning tolerance $\eta = 0.05$ leads to relatively low variance (RSD 0.15). Once again, $\eta = 0.025$ (RSD 0.23) achieves a well-balanced combination of metrics.

4.2 Local explanations

We analyze TimeSHAP's local explanations on 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

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

[3]A standardized measure of dispersion, computed as the ratio of the standard deviation to the mean, $\frac{\sigma}{\mu}$.

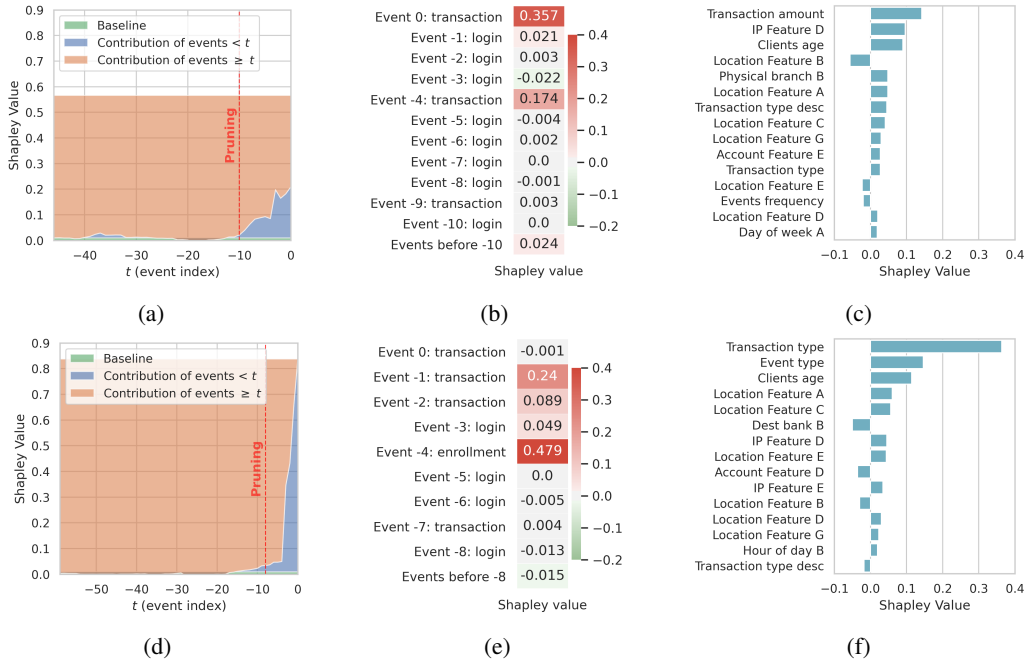

Figure 2: Figures (a), (b), and (c) show TimeSHAP results for Sequence A. Figures (d), (e), and (f) show TimeSHAP results for Sequence B. Figures (a) and (d) show the importance of older ($X_{:,:t-1}$) vs current events ($X_{:,t:}$) calculated using Algorithm 1 also displaying also Shaps' local accuracy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

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 older 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, when 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. When considering the 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.

Sequence A's event-wise explanations 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, there are three logins with little to no importance (events $-3 \leq t \leq -1$). Prior (in temporal order) to event $t = -4$, there are 5 logins and 1 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 both transactions $t = 0$ and $t = -4$ is unusually high, a known account takeover indicator. This is in accordance with the simultaneous high event importance for $t = 0$ and $t = -4$, 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 [45]. 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.

Sequence B's event-wise explanations are shown in Figure 2e, and its feature-wise explanations in Figure 2f. Regarding event importance, we conclude that the most relevant events are events at indices -4 and -1 with their respective 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 input ($t = 0$), 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, and 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 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.

4.3 Global explanations

Supplying a data scientist with global explanations can enable 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 model decision process guiding a data scientist analysis of the model. To obtain these global explanations, TimeSHAP is used to explain N sequences and drawing conclusions from aggregations and visualizations of the N local explanations. To obtain global explanations presented in this work, $1K$ randomly sampled positive-predicted sequences were explained using TimeSHAP and aggregations on both the event-wise and feature-wise

By analysing the global event-wise explanations, we conclude that the latest event ($t = 0$) has, on average, the highest event importance throughout each sequence, with an average Shapley value of 0.28. At the same time, we observe that events between indices -1 and -5 are often of high importance as well, with average contributions ranging from 0.03 to 0.12.

From analyzing global feature-wise explanations, we conclude that the most relevant features for this model, having a predominately positive contribution to the score (*i.e.*, serving as fraud indicators), are related to the transaction type (average contribution 0.28), the clients' age (0.092), the event type (0.09), and related to location and IP (0.08). These features are in accordance with our domain knowledge of account takeover, where the event and transaction type together with the location and IP features encode account behaviors, and the age of the client can, in most cases, indicate a possibly vulnerable accounts.

5 Conclusion

While considerable effort has been guided towards explaining deep learning models, recurrent models have received comparatively little attention. With this in mind, we presented TimeSHAP, a model-agnostic recurrent explainer that leverages KernelSHAP's sound theoretical footing and strong empirical results. TimeSHAP carries three main contributions: (1) a new perturbation function suited for the recurrent setting, (2) explanations of both feature- and event-wise importance, and (3) a sequence pruning method that dramatically increases the method's efficiency in practice.

TimeSHAP enables data scientists to see which features were most important for a model's prediction, but also which past events impact the current prediction the most. We apply our method on a large-scale real-world dataset on account takeover fraud. We notice potentially discriminatory reasoning based on the client's age, as it is shown to increase the model's fraud score on several occasions. Overall, we find multiple instances whose explanations are crucial for debugging the underlying predictor.

Acknowledgements

The project CAMELOT (reference POCI-01-0247-FEDER-045915) leading to this work is co-financed by the ERDF - European Regional Development Fund through the Operational Program for Competitiveness and Internationalisation - COMPETE 2020, the North Portugal Regional Operational Program - NORTE 2020 and by the Portuguese Foundation for Science and Technology - FCT under the CMU Portugal international partnership.

References

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

[2] 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.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2 : End-to-end speech recognition in english and mandarin. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 173–182, New York, New York, USA, 20–22 Jun 2016. PMLR.

[4] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier LSTM. In *International Conference on Learning Representations*, 2020.

[5] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, aug 2019.

[6] Oskar Pfungst and Carl Leo Rahn. *Clever Hans (the horse of Mr. Von Osten) a contribution to experimental animal and human psychology,*. H. Holt and company, New York,, 1911.

[7] 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.

[8] 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.

[9] 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.

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

[11] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`, May 2016. Accessed: 2020-01-09.

[12] Robert Bartlett, Adair Morse, Richard Stanton, and Nancy Wallace. Consumer-Lending Discrimination in the FinTech Era. Technical report, National Bureau of Economic Research, 2019.

[13] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In Sorelle A Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 2018. PMLR.

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

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

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

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

[39] 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.

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

[41] 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.

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

[43] 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.

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

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